

# EPICS-TwinCAT IOC

2.1

Generated by Doxygen 1.8.18



---

|  |           |
|--|-----------|
| <b>1 Module Index</b>                              | <b>1</b>  |
| 1.1 Modules . . . . .                              | 1         |
| <b>2 Namespace Index</b>                           | <b>3</b>  |
| 2.1 Namespace List . . . . .                       | 3         |
| <b>3 Hierarchical Index</b>                        | <b>5</b>  |
| 3.1 Class Hierarchy . . . . .                      | 5         |
| <b>4 Class Index</b>                               | <b>7</b>  |
| 4.1 Class List . . . . .                           | 7         |
| <b>5 File Index</b>                                | <b>11</b> |
| 5.1 File List . . . . .                            | 11        |
| <b>6 Module Documentation</b>                      | <b>13</b> |
| 6.1 Device support for TwinCAT/ADS . . . . .       | 13        |
| 6.1.1 Detailed Description . . . . .               | 16        |
| 6.1.2 Enumeration Type Documentation . . . . .     | 16        |
| 6.1.2.1 epics_record_enum . . . . .                | 16        |
| 6.1.3 Function Documentation . . . . .             | 17        |
| 6.1.3.1 get_callback_queue_free() . . . . .        | 17        |
| 6.1.3.2 get_callback_queue_size() . . . . .        | 18        |
| 6.1.3.3 get_callback_queue_used() . . . . .        | 18        |
| 6.1.3.4 linkRecord() . . . . .                     | 19        |
| 6.1.3.5 register_devsup() . . . . .                | 20        |
| 6.1.4 Variable Documentation . . . . .             | 20        |
| 6.1.4.1 isCallback . . . . .                       | 20        |
| 6.1.4.2 isPassive . . . . .                        | 21        |
| 6.1.4.3 the_register_devsup . . . . .              | 21        |
| 6.1.4.4 value_alt_type . . . . .                   | 21        |
| 6.1.4.5 value_count . . . . .                      | 21        |
| 6.2 Device driver functions . . . . .              | 22        |
| 6.2.1 Detailed Description . . . . .               | 22        |
| 6.3 Info interface classes and functions . . . . . | 23        |
| 6.3.1 Detailed Description . . . . .               | 23        |
| 6.3.2 Typedef Documentation . . . . .              | 23        |
| 6.3.2.1 info_dbrecord_type . . . . .               | 23        |
| 6.3.3 Enumeration Type Documentation . . . . .     | 23        |
| 6.3.3.1 update_enum . . . . .                      | 23        |
| 6.4 TwinCAT tpy file parser . . . . .              | 25        |
| 6.4.1 Detailed Description . . . . .               | 25        |
| 6.4.2 Typedef Documentation . . . . .              | 25        |
| 6.4.2.1 dimension . . . . .                        | 26        |

---

---

|  |    |
|--|----|
| 6.4.2.2 dimensions . . . . .                       | 26 |
| 6.4.2.3 enum_map . . . . .                         | 26 |
| 6.4.2.4 enum_pair . . . . .                        | 26 |
| 6.4.2.5 item_list . . . . .                        | 26 |
| 6.4.2.6 symbol_list . . . . .                      | 27 |
| 6.4.2.7 type_multimap . . . . .                    | 27 |
| 6.4.3 Enumeration Type Documentation . . . . .     | 27 |
| 6.4.3.1 type_enum . . . . .                        | 27 |
| 6.5 XML tpy file constants . . . . .               | 28 |
| 6.5.1 Detailed Description . . . . .               | 29 |
| 6.6 Parser utility functions and classes . . . . . | 30 |
| 6.6.1 Detailed Description . . . . .               | 30 |
| 6.6.2 Typedef Documentation . . . . .              | 30 |
| 6.6.2.1 property_el . . . . .                      | 31 |
| 6.6.2.2 property_map . . . . .                     | 31 |
| 6.6.2.3 replacement_table . . . . .                | 31 |
| 6.6.3 Enumeration Type Documentation . . . . .     | 31 |
| 6.6.3.1 opc_enum . . . . .                         | 31 |
| 6.6.3.2 process_tag_enum . . . . .                 | 32 |
| 6.6.3.3 process_type_enum . . . . .                | 32 |
| 6.7 OPC property constants . . . . .               | 33 |
| 6.7.1 Detailed Description . . . . .               | 34 |
| 6.7.2 Variable Documentation . . . . .             | 34 |
| 6.7.2.1 OPC_PROP_ALIAS . . . . .                   | 34 |
| 6.7.2.2 OPC_PROP_ALMAREAS . . . . .                | 34 |
| 6.7.2.3 OPC_PROP_ALMCONDITION . . . . .            | 34 |
| 6.7.2.4 OPC_PROP_ALMCOSV . . . . .                 | 35 |
| 6.7.2.5 OPC_PROP_ALMDB . . . . .                   | 35 |
| 6.7.2.6 OPC_PROP_ALMDEV . . . . .                  | 35 |
| 6.7.2.7 OPC_PROP_ALMFFSV . . . . .                 | 35 |
| 6.7.2.8 OPC_PROP_ALMH . . . . .                    | 36 |
| 6.7.2.9 OPC_PROP_ALMHELP . . . . .                 | 36 |
| 6.7.2.10 OPC_PROP_ALMH . . . . .                   | 36 |
| 6.7.2.11 OPC_PROP_ALMHHSV . . . . .                | 36 |
| 6.7.2.12 OPC_PROP_ALMHSV . . . . .                 | 37 |
| 6.7.2.13 OPC_PROP_ALML . . . . .                   | 37 |
| 6.7.2.14 OPC_PROP_ALMLIMIT . . . . .               | 37 |
| 6.7.2.15 OPC_PROP_ALMLL . . . . .                  | 37 |
| 6.7.2.16 OPC_PROP_ALMLLSV . . . . .                | 38 |
| 6.7.2.17 OPC_PROP_ALMLSV . . . . .                 | 38 |
| 6.7.2.18 OPC_PROP_ALMOSV . . . . .                 | 38 |
| 6.7.2.19 OPC_PROP_ALMPRIMARYAREA . . . . .         | 38 |

---

|  |    |
|--|----|
| 6.7.2.20 OPC_PROP_ALMROC . . . . .         | 38 |
| 6.7.2.21 OPC_PROP_ALMSTAT . . . . .        | 39 |
| 6.7.2.22 OPC_PROP_ALMUNSV . . . . .        | 39 |
| 6.7.2.23 OPC_PROP_ALMZRSV . . . . .        | 39 |
| 6.7.2.24 OPC_PROP_ALMZSV . . . . .         | 39 |
| 6.7.2.25 OPC_PROP_AVI . . . . .            | 39 |
| 6.7.2.26 OPC_PROP_BGC . . . . .            | 40 |
| 6.7.2.27 OPC_PROP_BLINK . . . . .          | 40 |
| 6.7.2.28 OPC_PROP_BMP . . . . .            | 40 |
| 6.7.2.29 OPC_PROP_CDT . . . . .            | 40 |
| 6.7.2.30 OPC_PROP_CLOSE . . . . .          | 40 |
| 6.7.2.31 OPC_PROP_DESC . . . . .           | 41 |
| 6.7.2.32 OPC_PROP_DTYP . . . . .           | 41 |
| 6.7.2.33 OPC_PROP_FFST . . . . .           | 41 |
| 6.7.2.34 OPC_PROP_FGC . . . . .            | 41 |
| 6.7.2.35 OPC_PROP_FIELD_BEG . . . . .      | 42 |
| 6.7.2.36 OPC_PROP_FIELD_END . . . . .      | 42 |
| 6.7.2.37 OPC_PROP_HIEU . . . . .           | 42 |
| 6.7.2.38 OPC_PROP_HIRANGE . . . . .        | 42 |
| 6.7.2.39 OPC_PROP_HTML . . . . .           | 43 |
| 6.7.2.40 OPC_PROP_INOUT . . . . .          | 43 |
| 6.7.2.41 OPC_PROP_INPUT . . . . .          | 43 |
| 6.7.2.42 OPC_PROP_LOEU . . . . .           | 43 |
| 6.7.2.43 OPC_PROP_LORANGE . . . . .        | 44 |
| 6.7.2.44 OPC_PROP_OPEN . . . . .           | 44 |
| 6.7.2.45 OPC_PROP_OUTPUT . . . . .         | 44 |
| 6.7.2.46 OPC_PROP_PINI . . . . .           | 44 |
| 6.7.2.47 OPC_PROP_PLNAME . . . . .         | 45 |
| 6.7.2.48 OPC_PROP_PREC . . . . .           | 45 |
| 6.7.2.49 OPC_PROP_QUALITY . . . . .        | 45 |
| 6.7.2.50 OPC_PROP_RECTYPE . . . . .        | 45 |
| 6.7.2.51 OPC_PROP_RIGHTS . . . . .         | 46 |
| 6.7.2.52 OPC_PROP_SCANRATE . . . . .       | 46 |
| 6.7.2.53 OPC_PROP_SERVER . . . . .         | 46 |
| 6.7.2.54 OPC_PROP SND . . . . .            | 46 |
| 6.7.2.55 OPC_PROP_TIME . . . . .           | 46 |
| 6.7.2.56 OPC_PROP_TIMEZONE . . . . .       | 47 |
| 6.7.2.57 OPC_PROP_TSE . . . . .            | 47 |
| 6.7.2.58 OPC_PROP_UNIT . . . . .           | 47 |
| 6.7.2.59 OPC_PROP_VALUE . . . . .          | 47 |
| 6.7.2.60 OPC_PROP_ZRST . . . . .           | 47 |
| 6.8 String functions and classes . . . . . | 48 |

---

---

|  |    |
|--|----|
| 6.8.1 Detailed Description . . . . .               | 48 |
| 6.8.2 Typedef Documentation . . . . .              | 48 |
| 6.8.2.1 stringcase . . . . .                       | 48 |
| 6.8.2.2 wstringcase . . . . .                      | 49 |
| 6.8.3 Function Documentation . . . . .             | 49 |
| 6.8.3.1 compare() [1/2] . . . . .                  | 49 |
| 6.8.3.2 compare() [2/2] . . . . .                  | 50 |
| 6.8.3.3 eq() [1/2] . . . . .                       | 50 |
| 6.8.3.4 eq() [2/2] . . . . .                       | 51 |
| 6.8.3.5 lt() [1/2] . . . . .                       | 51 |
| 6.8.3.6 lt() [2/2] . . . . .                       | 51 |
| 6.8.3.7 ne() [1/2] . . . . .                       | 52 |
| 6.8.3.8 ne() [2/2] . . . . .                       | 52 |
| 6.8.3.9 split_string() . . . . .                   | 52 |
| 6.8.3.10 strncasecmp() . . . . .                   | 53 |
| 6.8.3.11 trim_space() [1/2] . . . . .              | 54 |
| 6.8.3.12 trim_space() [2/2] . . . . .              | 55 |
| 6.8.3.13 wcsncasewcmp() . . . . .                  | 55 |
| 6.9 TwinCAT read/write scanning . . . . .          | 56 |
| 6.9.1 Detailed Description . . . . .               | 56 |
| 6.10 EPICS utility functions and classes . . . . . | 57 |
| 6.10.1 Detailed Description . . . . .              | 57 |
| 6.10.2 Typedef Documentation . . . . .             | 57 |
| 6.10.2.1 filename_set . . . . .                    | 58 |
| 6.10.2.2 macro_list . . . . .                      | 58 |
| 6.10.2.3 macro_stack . . . . .                     | 58 |
| 6.10.3 Enumeration Type Documentation . . . . .    | 58 |
| 6.10.3.1 case_type . . . . .                       | 58 |
| 6.10.3.2 device_support_type . . . . .             | 59 |
| 6.10.3.3 io_filestat . . . . .                     | 59 |
| 6.10.3.4 listing_type . . . . .                    | 59 |
| 6.10.3.5 macrofile_type . . . . .                  | 60 |
| 6.10.3.6 tc_epics_conv . . . . .                   | 60 |
| 6.11 EPICS constants . . . . .                     | 61 |
| 6.11.1 Detailed Description . . . . .              | 62 |
| 6.11.2 Variable Documentation . . . . .            | 62 |
| 6.11.2.1 EPICS_DB_ALLOWED . . . . .                | 62 |
| 6.11.2.2 EPICS_DB_COSV . . . . .                   | 62 |
| 6.11.2.3 EPICS_DB_DESC . . . . .                   | 62 |
| 6.11.2.4 EPICS_DB_DRVH . . . . .                   | 63 |
| 6.11.2.5 EPICS_DB_DRVL . . . . .                   | 63 |
| 6.11.2.6 EPICS_DB_DTYP . . . . .                   | 63 |

---

|  |    |
|--|----|
| 6.11.2.7 EPICS_DB_EGU . . . . .              | 63 |
| 6.11.2.8 EPICS_DB_FORBIDDEN . . . . .        | 64 |
| 6.11.2.9 EPICS_DB_HHSV . . . . .             | 64 |
| 6.11.2.10 EPICS_DB_HIGH . . . . .            | 64 |
| 6.11.2.11 EPICS_DB_HIHI . . . . .            | 64 |
| 6.11.2.12 EPICS_DB_HOPR . . . . .            | 65 |
| 6.11.2.13 EPICS_DB_HSV . . . . .             | 65 |
| 6.11.2.14 EPICS_DB_HYST . . . . .            | 65 |
| 6.11.2.15 EPICS_DB_INP . . . . .             | 65 |
| 6.11.2.16 EPICS_DB_LLSV . . . . .            | 66 |
| 6.11.2.17 EPICS_DB_LOLO . . . . .            | 66 |
| 6.11.2.18 EPICS_DB_LOPR . . . . .            | 66 |
| 6.11.2.19 EPICS_DB_LOW . . . . .             | 66 |
| 6.11.2.20 EPICS_DB_LSV . . . . .             | 67 |
| 6.11.2.21 EPICS_DB_MAJOR . . . . .           | 67 |
| 6.11.2.22 EPICS_DB_MINOR . . . . .           | 67 |
| 6.11.2.23 EPICS_DB_NOALARM . . . . .         | 67 |
| 6.11.2.24 EPICS_DB_NUMVAL . . . . .          | 68 |
| 6.11.2.25 EPICS_DB_ONAM . . . . .            | 68 |
| 6.11.2.26 EPICS_DB_OSV . . . . .             | 68 |
| 6.11.2.27 EPICS_DB_OUT . . . . .             | 68 |
| 6.11.2.28 EPICS_DB_PINI . . . . .            | 69 |
| 6.11.2.29 EPICS_DB_PREC . . . . .            | 69 |
| 6.11.2.30 EPICS_DB_SCAN . . . . .            | 69 |
| 6.11.2.31 EPICS_DB_TSE . . . . .             | 69 |
| 6.11.2.32 EPICS_DB_UNSV . . . . .            | 70 |
| 6.11.2.33 EPICS_DB_ZNAM . . . . .            | 70 |
| 6.11.2.34 EPICS_DB_ZRST . . . . .            | 70 |
| 6.11.2.35 EPICS_DB_ZRSV . . . . .            | 70 |
| 6.11.2.36 EPICS_DB_ZRVL . . . . .            | 71 |
| 6.11.2.37 EPICS_DB_ZSV . . . . .             | 71 |
| 6.11.2.38 LIGODAQ_DATATYPE_DEFAULT . . . . . | 71 |
| 6.11.2.39 LIGODAQ_DATATYPE_FLOAT . . . . .   | 71 |
| 6.11.2.40 LIGODAQ_DATATYPE_INT32 . . . . .   | 72 |
| 6.11.2.41 LIGODAQ_DATATYPE_NAME . . . . .    | 72 |
| 6.11.2.42 LIGODAQ_INI_HEADER . . . . .       | 72 |
| 6.11.2.43 LIGODAQ_UNIT_DEFAULT . . . . .     | 72 |
| 6.11.2.44 LIGODAQ_UNIT_NAME . . . . .        | 73 |
| 6.11.2.45 LIGODAQ_UNIT_NONE . . . . .        | 73 |
| 6.11.2.46 MAX_EPICS_CHANNEL . . . . .        | 73 |
| 6.11.2.47 MAX_EPICS_DESC . . . . .           | 73 |
| 6.11.2.48 MAX_EPICS_STRING . . . . .         | 73 |

---

|  |           |
|--|-----------|
| 6.11.2.49 MAX_EPICS_UNIT . . . . .             | 73        |
| <b>7 Namespace Documentation</b>               | <b>75</b> |
| 7.1 DevTc Namespace Reference . . . . .        | 75        |
| 7.1.1 Detailed Description . . . . .           | 76        |
| 7.1.2 Function Documentation . . . . .         | 76        |
| 7.1.2.1 tc_regex() . . . . .                   | 77        |
| 7.1.2.2 tcAlias() . . . . .                    | 77        |
| 7.1.2.3 tcInfoPrefix() . . . . .               | 78        |
| 7.1.2.4 tcList() . . . . .                     | 78        |
| 7.1.2.5 tcLoadRecords() . . . . .              | 79        |
| 7.1.2.6 tcMacro() . . . . .                    | 79        |
| 7.1.2.7 tcPrintVal() . . . . .                 | 80        |
| 7.1.2.8 tcPrintVals() . . . . .                | 81        |
| 7.1.2.9 tcSetScanRate() . . . . .              | 81        |
| 7.2 EpicsTpy Namespace Reference . . . . .     | 82        |
| 7.2.1 Detailed Description . . . . .           | 84        |
| 7.3 InfoPlc Namespace Reference . . . . .      | 84        |
| 7.3.1 Detailed Description . . . . .           | 85        |
| 7.4 ParseTpy Namespace Reference . . . . .     | 85        |
| 7.4.1 Detailed Description . . . . .           | 87        |
| 7.4.2 Function Documentation . . . . .         | 87        |
| 7.4.2.1 compareNamesWoNamespace() . . . . .    | 88        |
| 7.4.2.2 get_decoration() . . . . .             | 88        |
| 7.4.2.3 get_pointer() . . . . .                | 88        |
| 7.5 ParseUtil Namespace Reference . . . . .    | 88        |
| 7.5.1 Detailed Description . . . . .           | 91        |
| 7.6 plc Namespace Reference . . . . .          | 91        |
| 7.6.1 Detailed Description . . . . .           | 92        |
| 7.6.2 Typedef Documentation . . . . .          | 92        |
| 7.6.2.1 BasePLCList . . . . .                  | 93        |
| 7.6.2.2 BasePLCPtr . . . . .                   | 93        |
| 7.6.2.3 BaseRecordList . . . . .               | 93        |
| 7.6.2.4 BaseRecordPtr . . . . .                | 93        |
| 7.6.2.5 InterfacePtr . . . . .                 | 94        |
| 7.6.3 Enumeration Type Documentation . . . . . | 94        |
| 7.6.3.1 access_rights_enum . . . . .           | 94        |
| 7.6.3.2 data_type_enum . . . . .               | 95        |
| 7.6.4 Function Documentation . . . . .         | 95        |
| 7.6.4.1 reset_and_read() [1/3] . . . . .       | 95        |
| 7.6.4.2 reset_and_read() [2/3] . . . . .       | 96        |
| 7.6.4.3 reset_and_read() [3/3] . . . . .       | 96        |

---

---

|  |            |
|--|------------|
| 7.6.4.4 ScannerProc()                                      | 97         |
| 7.6.4.5 scannerThread()                                    | 98         |
| 7.6.4.6 write_and_test() [1/3]                             | 99         |
| 7.6.4.7 write_and_test() [2/3]                             | 99         |
| 7.6.4.8 write_and_test() [3/3]                             | 100        |
| 7.7 TcComms Namespace Reference                            | 100        |
| 7.7.1 Detailed Description                                 | 101        |
| 7.7.2 Function Documentation                               | 101        |
| 7.7.2.1 ADScallback()                                      | 101        |
| 7.7.2.2 errorPrintf()                                      | 102        |
| 7.7.2.3 RouterCall()                                       | 102        |
| <b>8 Class Documentation</b>                               | <b>103</b> |
| 8.1 ParseTpy::ads_routing_info Class Reference             | 103        |
| 8.1.1 Detailed Description                                 | 104        |
| 8.1.2 Member Function Documentation                        | 104        |
| 8.1.2.1 get() [1/2]  | 104        |
| 8.1.2.2 get() [2/2]  | 105        |
| 8.1.2.3 set()  | 105        |
| 8.2 TcComms::AmsRouterNotification Class Reference         | 106        |
| 8.2.1 Detailed Description                                 | 107        |
| 8.2.2 Friends And Related Function Documentation           | 107        |
| 8.2.2.1 RouterCall   | 107        |
| 8.3 std::atomic< string > Class Reference                  | 107        |
| 8.3.1 Detailed Description                                 | 108        |
| 8.4 std::atomic< wstring > Class Reference                 | 108        |
| 8.4.1 Detailed Description                                 | 109        |
| 8.5 std::atomic_string< stringT > Class Template Reference | 110        |
| 8.5.1 Detailed Description                                 | 110        |
| 8.5.2 Member Function Documentation                        | 111        |
| 8.5.2.1 operator=()  | 111        |
| 8.6 ParseTpy::base_record Class Reference                  | 111        |
| 8.6.1 Detailed Description                                 | 112        |
| 8.6.2 Constructor & Destructor Documentation               | 113        |
| 8.6.2.1 base_record() [1/4]                                | 113        |
| 8.6.2.2 base_record() [2/4]                                | 113        |
| 8.6.2.3 base_record() [3/4]                                | 113        |
| 8.6.2.4 base_record() [4/4]                                | 114        |
| 8.7 plc::BasePLC Class Reference                           | 114        |
| 8.7.1 Detailed Description                                 | 117        |
| 8.7.2 Member Function Documentation                        | 117        |
| 8.7.2.1 add() [1/2]  | 117        |

|  |     |
|--|-----|
| 8.7.2.2 add() [2/2] . . . . .                          | 118 |
| 8.7.2.3 erase() . . . . .                              | 119 |
| 8.7.2.4 find() . . . . .                               | 119 |
| 8.7.2.5 for_each() . . . . .                           | 120 |
| 8.7.2.6 get_next() . . . . .                           | 121 |
| 8.7.2.7 get_timestamp_unix() . . . . .                 | 121 |
| 8.7.2.8 plc_data_set_valid() . . . . .                 | 122 |
| 8.7.2.9 printRecord() . . . . .                        | 123 |
| 8.7.2.10 reserve() . . . . .                           | 123 |
| 8.7.2.11 start() . . . . .                             | 123 |
| 8.7.2.12 user_data_set_valid() . . . . .               | 124 |
| 8.7.3 Member Data Documentation . . . . .              | 124 |
| 8.7.3.1 records . . . . .                              | 124 |
| 8.8 plc::BaseRecord Class Reference . . . . .          | 125 |
| 8.8.1 Detailed Description . . . . .                   | 127 |
| 8.8.2 Constructor & Destructor Documentation . . . . . | 128 |
| 8.8.2.1 BaseRecord() [1/2] . . . . .                   | 128 |
| 8.8.2.2 BaseRecord() [2/2] . . . . .                   | 128 |
| 8.8.3 Member Function Documentation . . . . .          | 128 |
| 8.8.3.1 PlcGetValid() . . . . .                        | 129 |
| 8.8.3.2 PlcPush() . . . . .                            | 129 |
| 8.8.3.3 PlcRead() [1/3] . . . . .                      | 130 |
| 8.8.3.4 PlcRead() [2/3] . . . . .                      | 131 |
| 8.8.3.5 PlcRead() [3/3] . . . . .                      | 131 |
| 8.8.3.6 PlcReadBinary() . . . . .                      | 132 |
| 8.8.3.7 PlcSetValid() . . . . .                        | 133 |
| 8.8.3.8 PlcWrite() [1/3] . . . . .                     | 134 |
| 8.8.3.9 PlcWrite() [2/3] . . . . .                     | 135 |
| 8.8.3.10 PlcWrite() [3/3] . . . . .                    | 135 |
| 8.8.3.11 PlcWriteBinary() . . . . .                    | 136 |
| 8.8.3.12 UserGetValid() . . . . .                      | 137 |
| 8.8.3.13 UserPush() . . . . .                          | 137 |
| 8.8.3.14 UserRead() [1/3] . . . . .                    | 139 |
| 8.8.3.15 UserRead() [2/3] . . . . .                    | 140 |
| 8.8.3.16 UserRead() [3/3] . . . . .                    | 141 |
| 8.8.3.17 UserReadBinary() . . . . .                    | 142 |
| 8.8.3.18 UserSetValid() . . . . .                      | 142 |
| 8.8.3.19 UserWrite() [1/3] . . . . .                   | 143 |
| 8.8.3.20 UserWrite() [2/3] . . . . .                   | 144 |
| 8.8.3.21 UserWrite() [3/3] . . . . .                   | 145 |
| 8.8.3.22 UserWriteBinary() . . . . .                   | 145 |
| 8.9 ParseUtil::bit_location Class Reference . . . . .  | 146 |

---

---

|   |     |
|---|-----|
| 8.9.1 Detailed Description . . . . .                    | 147 |
| 8.10 std::case_char_traits Struct Reference . . . . .   | 147 |
| 8.10.1 Detailed Description . . . . .                   | 148 |
| 8.11 std::case_wchar_traits Struct Reference . . . . .  | 149 |
| 8.11.1 Detailed Description . . . . .                   | 149 |
| 8.12 ParseTpy::compiler_info Class Reference . . . . .  | 150 |
| 8.12.1 Detailed Description . . . . .                   | 151 |
| 8.13 TcComms::DataPar Struct Reference . . . . .        | 151 |
| 8.13.1 Detailed Description . . . . .                   | 152 |
| 8.14 plc::DataValue Class Reference . . . . .           | 152 |
| 8.14.1 Detailed Description . . . . .                   | 155 |
| 8.14.2 Constructor & Destructor Documentation . . . . . | 155 |
| 8.14.2.1 DataValue() . . . . .                          | 155 |
| 8.14.3 Member Function Documentation . . . . .          | 156 |
| 8.14.3.1 GetValid() . . . . .                           | 156 |
| 8.14.3.2 Init() . . . . .                               | 156 |
| 8.14.3.3 PlcGetValid() . . . . .                        | 157 |
| 8.14.3.4 PlcRead() [1/4] . . . . .                      | 158 |
| 8.14.3.5 PlcRead() [2/4] . . . . .                      | 159 |
| 8.14.3.6 PlcRead() [3/4] . . . . .                      | 159 |
| 8.14.3.7 PlcRead() [4/4] . . . . .                      | 160 |
| 8.14.3.8 PlcReadBinary() . . . . .                      | 160 |
| 8.14.3.9 PlcSetValid() . . . . .                        | 161 |
| 8.14.3.10 PlcWrite() [1/4] . . . . .                    | 162 |
| 8.14.3.11 PlcWrite() [2/4] . . . . .                    | 163 |
| 8.14.3.12 PlcWrite() [3/4] . . . . .                    | 164 |
| 8.14.3.13 PlcWrite() [4/4] . . . . .                    | 165 |
| 8.14.3.14 PlcWriteBinary() . . . . .                    | 165 |
| 8.14.3.15 Read() [1/5] . . . . .                        | 166 |
| 8.14.3.16 Read() [2/5] . . . . .                        | 167 |
| 8.14.3.17 Read() [3/5] . . . . .                        | 167 |
| 8.14.3.18 Read() [4/5] . . . . .                        | 169 |
| 8.14.3.19 Read() [5/5] . . . . .                        | 170 |
| 8.14.3.20 ReadBinary() . . . . .                        | 170 |
| 8.14.3.21 SetValid() . . . . .                          | 171 |
| 8.14.3.22 UserGetValid() . . . . .                      | 172 |
| 8.14.3.23 UserRead() [1/4] . . . . .                    | 173 |
| 8.14.3.24 UserRead() [2/4] . . . . .                    | 173 |
| 8.14.3.25 UserRead() [3/4] . . . . .                    | 174 |
| 8.14.3.26 UserRead() [4/4] . . . . .                    | 174 |
| 8.14.3.27 UserReadBinary() . . . . .                    | 175 |
| 8.14.3.28 UserSetValid() . . . . .                      | 176 |

---

|   |     |
|---|-----|
| 8.14.3.29 UserWrite() [1/4] . . . . .   | 177 |
| 8.14.3.30 UserWrite() [2/4] . . . . .   | 177 |
| 8.14.3.31 UserWrite() [3/4] . . . . .   | 178 |
| 8.14.3.32 UserWrite() [4/4] . . . . .   | 178 |
| 8.14.3.33 UserWriteBinary() . . . . .   | 179 |
| 8.14.3.34 Write() [1/5] . . . . .   | 180 |
| 8.14.3.35 Write() [2/5] . . . . .   | 181 |
| 8.14.3.36 Write() [3/5] . . . . .   | 182 |
| 8.14.3.37 Write() [4/5] . . . . .   | 183 |
| 8.14.3.38 Write() [5/5] . . . . .   | 183 |
| 8.14.3.39 WriteBinary() . . . . .   | 184 |
| 8.14.4 Member Data Documentation . . . . .                                    | 185 |
| 8.14.4.1 mysize . . . . .   | 185 |
| 8.15 plc::DataValueTraits< T > Struct Template Reference . . . . .            | 185 |
| 8.15.1 Detailed Description . . . . .   | 186 |
| 8.15.2 Member Data Documentation . . . . .                                    | 187 |
| 8.15.2.1 data_enum . . . . .  | 187 |
| 8.16 plc::DataValueTypeDef Struct Reference . . . . .                         | 187 |
| 8.16.1 Detailed Description . . . . .   | 189 |
| 8.17 DevTc::devTcDefIn< RecType > Struct Template Reference . . . . .         | 189 |
| 8.17.1 Detailed Description . . . . .   | 191 |
| 8.18 DevTc::devTcDeflo< RecType > Struct Template Reference . . . . .         | 191 |
| 8.18.1 Detailed Description . . . . .   | 192 |
| 8.19 DevTc::devTcDefOut< RecType > Struct Template Reference . . . . .        | 193 |
| 8.19.1 Detailed Description . . . . .   | 194 |
| 8.20 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference . . . . . | 194 |
| 8.20.1 Detailed Description . . . . .   | 195 |
| 8.21 EpicsTpy::epics_conversion Class Reference . . . . .                     | 196 |
| 8.21.1 Detailed Description . . . . .   | 197 |
| 8.21.2 Constructor & Destructor Documentation . . . . .                       | 197 |
| 8.21.2.1 epics_conversion() [1/3] . . . . .                                   | 197 |
| 8.21.2.2 epics_conversion() [2/3] . . . . .                                   | 198 |
| 8.21.2.3 epics_conversion() [3/3] . . . . .                                   | 198 |
| 8.21.3 Member Function Documentation . . . . .                                | 199 |
| 8.21.3.1 getopt() . . . . .   | 199 |
| 8.21.3.2 to_epics() . . . . .   | 200 |
| 8.22 EpicsTpy::epics_db_processing Class Reference . . . . .                  | 201 |
| 8.22.1 Detailed Description . . . . .   | 203 |
| 8.22.2 Constructor & Destructor Documentation . . . . .                       | 203 |
| 8.22.2.1 epics_db_processing() . . . . .                                      | 203 |
| 8.22.3 Member Function Documentation . . . . .                                | 204 |
| 8.22.3.1 getopt() . . . . .   | 204 |

---

---

|  |     |
|--|-----|
| 8.22.3.2 my getopt() . . . . .   | 205 |
| 8.22.3.3 operator()() . . . . .  | 206 |
| 8.22.3.4 process_field_alarm() . . . . .                                       | 208 |
| 8.22.3.5 process_field_numeric() [1/3] . . . . .                               | 209 |
| 8.22.3.6 process_field_numeric() [2/3] . . . . .                               | 210 |
| 8.22.3.7 process_field_numeric() [3/3] . . . . .                               | 211 |
| 8.22.3.8 process_field_string() . . . . .                                      | 212 |
| 8.23 EpicsTpy::epics_list_processing Class Reference . . . . .                 | 213 |
| 8.23.1 Detailed Description . . . . .  | 215 |
| 8.23.2 Constructor & Destructor Documentation . . . . .                        | 215 |
| 8.23.2.1 epics_list_processing() [1/2] . . . . .                               | 215 |
| 8.23.2.2 epics_list_processing() [2/2] . . . . .                               | 215 |
| 8.23.3 Member Function Documentation . . . . .                                 | 216 |
| 8.23.3.1 getopt() . . . . .  | 216 |
| 8.23.3.2 my getopt() . . . . .   | 217 |
| 8.23.3.3 operator()() . . . . .  | 218 |
| 8.24 EpicsTpy::epics_macrofiles_processing Class Reference . . . . .           | 220 |
| 8.24.1 Detailed Description . . . . .  | 222 |
| 8.24.2 Constructor & Destructor Documentation . . . . .                        | 222 |
| 8.24.2.1 epics_macrofiles_processing() [1/2] . . . . .                         | 222 |
| 8.24.2.2 epics_macrofiles_processing() [2/2] . . . . .                         | 222 |
| 8.24.3 Member Function Documentation . . . . .                                 | 223 |
| 8.24.3.1 getopt() . . . . .  | 223 |
| 8.24.3.2 my getopt() . . . . .   | 224 |
| 8.24.3.3 operator()() . . . . .  | 225 |
| 8.24.3.4 to_filename() . . . . .   | 226 |
| 8.25 DevTc::epics_record_traits< RecType > Struct Template Reference . . . . . | 227 |
| 8.25.1 Detailed Description . . . . .  | 228 |
| 8.26 DevTc::epics_tc_db_processing Class Reference . . . . .                   | 228 |
| 8.26.1 Detailed Description . . . . .  | 230 |
| 8.26.2 Constructor & Destructor Documentation . . . . .                        | 230 |
| 8.26.2.1 epics_tc_db_processing() . . . . .                                    | 230 |
| 8.26.3 Member Function Documentation . . . . .                                 | 230 |
| 8.26.3.1 operator()() . . . . .  | 231 |
| 8.26.3.2 patch_db_recordnames() . . . . .                                      | 231 |
| 8.26.3.3 process_list() . . . . .  | 231 |
| 8.26.3.4 process_lists() . . . . .   | 232 |
| 8.26.3.5 process_macro() . . . . .   | 232 |
| 8.26.3.6 process_macros() . . . . .  | 232 |
| 8.27 DevTc::EpicsInterface Class Reference . . . . .                           | 233 |
| 8.27.1 Detailed Description . . . . .  | 236 |
| 8.28 std::std::hash< std::stringcase > Struct Reference . . . . .              | 236 |

---

---

|  |     |
|--|-----|
| 8.28.1 Detailed Description . . . . .                              | 236 |
| 8.28.2 Member Function Documentation . . . . .                     | 236 |
| 8.28.2.1 operator()() . . . . .                                    | 236 |
| 8.29 std::std::hash< std::wstringcase > Struct Reference . . . . . | 237 |
| 8.29.1 Detailed Description . . . . .                              | 237 |
| 8.29.2 Member Function Documentation . . . . .                     | 237 |
| 8.29.2.1 operator()() . . . . .                                    | 237 |
| 8.30 InfoPlc::InfoInterface Class Reference . . . . .              | 238 |
| 8.30.1 Detailed Description . . . . .                              | 242 |
| 8.30.2 Constructor & Destructor Documentation . . . . .            | 242 |
| 8.30.2.1 InfoInterface() [1/2] . . . . .                           | 242 |
| 8.30.2.2 InfoInterface() [2/2] . . . . .                           | 243 |
| 8.30.3 Member Function Documentation . . . . .                     | 243 |
| 8.30.3.1 get_infodb() . . . . .                                    | 243 |
| 8.30.3.2 get_symbol_name() . . . . .                               | 244 |
| 8.30.3.3 printVal() . . . . .                                      | 244 |
| 8.30.4 Member Data Documentation . . . . .                         | 245 |
| 8.30.4.1 dbinfo_list . . . . .                                     | 245 |
| 8.31 plc::Interface Class Reference . . . . .                      | 245 |
| 8.31.1 Detailed Description . . . . .                              | 247 |
| 8.31.2 Constructor & Destructor Documentation . . . . .            | 247 |
| 8.31.2.1 Interface() . . . . .                                     | 247 |
| 8.31.3 Member Function Documentation . . . . .                     | 247 |
| 8.31.3.1 printVal() . . . . .                                      | 247 |
| 8.32 ParseTpy::item_record Class Reference . . . . .               | 248 |
| 8.32.1 Detailed Description . . . . .                              | 249 |
| 8.33 EpicsTpy::macro_info Struct Reference . . . . .               | 249 |
| 8.33.1 Detailed Description . . . . .                              | 250 |
| 8.34 EpicsTpy::macro_record Struct Reference . . . . .             | 250 |
| 8.34.1 Detailed Description . . . . .                              | 251 |
| 8.35 ParseUtil::memory_location Class Reference . . . . .          | 251 |
| 8.35.1 Detailed Description . . . . .                              | 252 |
| 8.35.2 Constructor & Destructor Documentation . . . . .            | 252 |
| 8.35.2.1 memory_location() [1/2] . . . . .                         | 252 |
| 8.35.2.2 memory_location() [2/2] . . . . .                         | 253 |
| 8.35.3 Member Function Documentation . . . . .                     | 253 |
| 8.35.3.1 get() . . . . .   | 253 |
| 8.35.3.2 set() . . . . .   | 254 |
| 8.35.3.3 set_section() . . . . .                                   | 255 |
| 8.36 EpicsTpy::multi_io_support Class Reference . . . . .          | 255 |
| 8.36.1 Detailed Description . . . . .                              | 257 |
| 8.36.2 Constructor & Destructor Documentation . . . . .            | 257 |

---

---

|   |     |
|---|-----|
| 8.36.2.1 <code>multi_io_support()</code>          | 257 |
| 8.36.3 Member Function Documentation              | 258 |
| 8.36.3.1 <code>getopt()</code>                    | 258 |
| 8.37 ParseUtil::opc_list Class Reference          | 259 |
| 8.37.1 Detailed Description                       | 260 |
| 8.38 ParseUtil::optarg Class Reference            | 260 |
| 8.38.1 Detailed Description                       | 261 |
| 8.38.2 Constructor & Destructor Documentation     | 262 |
| 8.38.2.1 <code>optarg()</code>                    | 262 |
| 8.38.3 Member Function Documentation              | 262 |
| 8.38.3.1 <code>parse()</code>                     | 262 |
| 8.39 ParseTpy::parserinfo_type Class Reference    | 263 |
| 8.39.1 Detailed Description                       | 265 |
| 8.39.2 Member Data Documentation                  | 266 |
| 8.39.2.1 <code>name_parse</code>                  | 266 |
| 8.40 ParseUtil::process_arg Class Reference       | 266 |
| 8.40.1 Detailed Description                       | 267 |
| 8.40.2 Constructor & Destructor Documentation     | 267 |
| 8.40.2.1 <code>process_arg()</code>               | 268 |
| 8.40.3 Member Function Documentation              | 269 |
| 8.40.3.1 <code>get_full()</code>                  | 269 |
| 8.41 InfoPlc::process_arg_info Class Reference    | 270 |
| 8.41.1 Detailed Description                       | 271 |
| 8.41.2 Constructor & Destructor Documentation     | 271 |
| 8.41.2.1 <code>process_arg_info()</code>          | 271 |
| 8.41.3 Member Function Documentation              | 271 |
| 8.41.3.1 <code>get_full()</code>                  | 271 |
| 8.42 ParseUtil::process_arg_tc Class Reference    | 272 |
| 8.42.1 Detailed Description                       | 273 |
| 8.42.2 Constructor & Destructor Documentation     | 273 |
| 8.42.2.1 <code>process_arg_tc()</code>            | 274 |
| 8.42.3 Member Function Documentation              | 274 |
| 8.42.3.1 <code>get()</code>                       | 274 |
| 8.42.3.2 <code>get_full()</code>                  | 275 |
| 8.43 ParseTpy::project_record Class Reference     | 275 |
| 8.43.1 Detailed Description                       | 276 |
| 8.44 DevTc::register_devsup Class Reference       | 276 |
| 8.44.1 Detailed Description                       | 277 |
| 8.45 ParseUtil::replacement_rules Class Reference | 278 |
| 8.45.1 Detailed Description                       | 279 |
| 8.46 plc::scanner_thread_args Struct Reference    | 279 |
| 8.46.1 Detailed Description                       | 280 |

---

---

|   |     |
|---|-----|
| 8.47 EpicsTpy::split_io_support Class Reference . . . . . | 280 |
| 8.47.1 Detailed Description . . . . .                     | 282 |
| 8.47.2 Constructor & Destructor Documentation . . . . .   | 282 |
| 8.47.2.1 split_io_support() [1/2] . . . . .               | 282 |
| 8.47.2.2 split_io_support() [2/2] . . . . .               | 283 |
| 8.47.3 Member Function Documentation . . . . .            | 283 |
| 8.47.3.1 getopt() . . . . .                               | 283 |
| 8.47.3.2 increment() . . . . .                            | 284 |
| 8.47.3.3 operator=() . . . . .                            | 285 |
| 8.48 ParseTpy::symbol_record Class Reference . . . . .    | 286 |
| 8.48.1 Detailed Description . . . . .                     | 287 |
| 8.49 syminfo_processing Class Reference . . . . .         | 287 |
| 8.49.1 Detailed Description . . . . .                     | 287 |
| 8.50 plc::System Class Reference . . . . .                | 288 |
| 8.50.1 Detailed Description . . . . .                     | 289 |
| 8.50.2 Member Function Documentation . . . . .            | 289 |
| 8.50.2.1 add() [1/2] . . . . .                            | 289 |
| 8.50.2.2 add() [2/2] . . . . .                            | 289 |
| 8.50.2.3 for_each() . . . . .                             | 290 |
| 8.50.2.4 printVal() . . . . .                             | 290 |
| 8.51 ParseUtil::tag_processing Class Reference . . . . .  | 291 |
| 8.51.1 Detailed Description . . . . .                     | 292 |
| 8.51.2 Constructor & Destructor Documentation . . . . .   | 292 |
| 8.51.2.1 tag_processing() [1/2] . . . . .                 | 292 |
| 8.51.2.2 tag_processing() [2/2] . . . . .                 | 293 |
| 8.51.3 Member Function Documentation . . . . .            | 293 |
| 8.51.3.1 getopt() . . . . .                               | 293 |
| 8.52 TcComms::TCatInterface Class Reference . . . . .     | 295 |
| 8.52.1 Detailed Description . . . . .                     | 297 |
| 8.52.2 Constructor & Destructor Documentation . . . . .   | 297 |
| 8.52.2.1 TCatInterface() . . . . .                        | 297 |
| 8.52.3 Member Function Documentation . . . . .            | 298 |
| 8.52.3.1 printVal() . . . . .                             | 298 |
| 8.53 TcComms::TcPLC Class Reference . . . . .             | 299 |
| 8.53.1 Detailed Description . . . . .                     | 302 |
| 8.53.2 Member Function Documentation . . . . .            | 302 |
| 8.53.2.1 closePort() . . . . .                            | 302 |
| 8.53.2.2 get_responseBuffer() . . . . .                   | 302 |
| 8.53.2.3 optimizeRequests() . . . . .                     | 303 |
| 8.53.2.4 printRecord() . . . . .                          | 303 |
| 8.53.2.5 set_addr() . . . . .                             | 304 |
| 8.53.2.6 set_ads_state() . . . . .                        | 305 |

---

---

|  |            |
|--|------------|
| 8.53.2.7 update_scanner() . . . . .  | 305        |
| 8.53.3 Friends And Related Function Documentation . . . . .                        | 306        |
| 8.53.3.1 ADSCallback . . . . .   | 306        |
| 8.53.4 Member Data Documentation . . . . .   | 306        |
| 8.53.4.1 cyclesLeft . . . . .  | 306        |
| 8.54 TcComms::tcProcWrite Class Reference . . . . .                                | 307        |
| 8.54.1 Detailed Description . . . . .  | 308        |
| 8.54.2 Member Function Documentation . . . . .                                     | 308        |
| 8.54.2.1 add() . . . . .   | 308        |
| 8.54.2.2 read_ptr() . . . . .  | 309        |
| 8.54.3 Member Data Documentation . . . . .   | 310        |
| 8.54.3.1 req . . . . .   | 310        |
| 8.55 DevTc::tcRegisterTolocShell Class Reference . . . . .                         | 310        |
| 8.55.1 Detailed Description . . . . .  | 310        |
| 8.56 ParseTpy::tpy_file Class Reference . . . . .                                  | 311        |
| 8.56.1 Detailed Description . . . . .  | 312        |
| 8.56.2 Member Function Documentation . . . . .                                     | 312        |
| 8.56.2.1 parse_finish() . . . . .  | 313        |
| 8.56.2.2 process_array() . . . . .   | 313        |
| 8.56.2.3 process_symbols() . . . . .   | 314        |
| 8.56.2.4 process_type_tree() [1/3] . . . . .                                       | 316        |
| 8.56.2.5 process_type_tree() [2/3] . . . . .                                       | 317        |
| 8.56.2.6 process_type_tree() [3/3] . . . . .                                       | 319        |
| 8.57 DevTc::epics_record_traits< RecType >::traits_type Struct Reference . . . . . | 320        |
| 8.57.1 Detailed Description . . . . .  | 321        |
| 8.58 ParseTpy::type_map Class Reference . . . . .                                  | 321        |
| 8.58.1 Detailed Description . . . . .  | 322        |
| 8.59 ParseTpy::type_record Class Reference . . . . .                               | 322        |
| 8.59.1 Detailed Description . . . . .  | 324        |
| 8.60 ParseUtil::variable_name Class Reference . . . . .                            | 324        |
| 8.60.1 Detailed Description . . . . .  | 325        |
| <b>9 File Documentation</b> . . . . .  | <b>327</b> |
| 9.1 atomic_string.h File Reference . . . . .                                       | 327        |
| 9.1.1 Detailed Description . . . . .   | 328        |
| 9.2 devTc.cpp File Reference . . . . .   | 328        |
| 9.2.1 Detailed Description . . . . .   | 330        |
| 9.3 devTc.h File Reference . . . . .   | 330        |
| 9.3.1 Detailed Description . . . . .   | 332        |
| 9.4 drvTc.cpp File Reference . . . . .   | 332        |
| 9.4.1 Detailed Description . . . . .   | 333        |
| 9.5 drvTc.h File Reference . . . . .   | 334        |

---

---

|  |     |
|--|-----|
| 9.5.1 Detailed Description . . . . .             | 334 |
| 9.6 EpicsDbGen.cpp File Reference . . . . .      | 335 |
| 9.6.1 Detailed Description . . . . .             | 335 |
| 9.6.2 Function Documentation . . . . .           | 335 |
| 9.6.2.1 main() . . . . .                         | 335 |
| 9.7 infoPlc.cpp File Reference . . . . .         | 336 |
| 9.7.1 Detailed Description . . . . .             | 337 |
| 9.8 infoPlc.h File Reference . . . . .           | 337 |
| 9.8.1 Detailed Description . . . . .             | 338 |
| 9.9 infoPlcTemplate.h File Reference . . . . .   | 338 |
| 9.9.1 Detailed Description . . . . .             | 339 |
| 9.10 iocMain.cpp File Reference . . . . .        | 339 |
| 9.10.1 Detailed Description . . . . .            | 340 |
| 9.10.2 Function Documentation . . . . .          | 340 |
| 9.10.2.1 main() . . . . .                        | 340 |
| 9.11 ParseTpy.cpp File Reference . . . . .       | 340 |
| 9.11.1 Detailed Description . . . . .            | 341 |
| 9.12 ParseTpy.h File Reference . . . . .         | 341 |
| 9.12.1 Detailed Description . . . . .            | 342 |
| 9.13 ParseTpyConst.h File Reference . . . . .    | 343 |
| 9.13.1 Detailed Description . . . . .            | 345 |
| 9.14 ParseTpyInfo.cpp File Reference . . . . .   | 345 |
| 9.14.1 Detailed Description . . . . .            | 345 |
| 9.14.2 Function Documentation . . . . .          | 346 |
| 9.14.2.1 main() . . . . .                        | 346 |
| 9.15 ParseTpyTemplate.h File Reference . . . . . | 347 |
| 9.15.1 Detailed Description . . . . .            | 347 |
| 9.16 ParseUtil.cpp File Reference . . . . .      | 347 |
| 9.16.1 Detailed Description . . . . .            | 348 |
| 9.17 ParseUtil.h File Reference . . . . .        | 348 |
| 9.17.1 Detailed Description . . . . .            | 349 |
| 9.18 ParseUtilConst.h File Reference . . . . .   | 349 |
| 9.18.1 Detailed Description . . . . .            | 351 |
| 9.19 plcBase.cpp File Reference . . . . .        | 351 |
| 9.19.1 Detailed Description . . . . .            | 352 |
| 9.20 plcBase.h File Reference . . . . .          | 352 |
| 9.20.1 Detailed Description . . . . .            | 354 |
| 9.21 plcBaseTemplate.h File Reference . . . . .  | 355 |
| 9.21.1 Detailed Description . . . . .            | 356 |
| 9.22 stdafx.cpp File Reference . . . . .         | 356 |
| 9.22.1 Detailed Description . . . . .            | 356 |
| 9.23 stdafx.h File Reference . . . . .           | 356 |

|   |            |
|---|------------|
| 9.23.1 Detailed Description . . . . .           | 357        |
| 9.24 stringcase.h File Reference . . . . .      | 357        |
| 9.24.1 Detailed Description . . . . .           | 358        |
| 9.25 stringcase_hash.h File Reference . . . . . | 358        |
| 9.25.1 Detailed Description . . . . .           | 359        |
| 9.26 tcComms.cpp File Reference . . . . .       | 359        |
| 9.26.1 Detailed Description . . . . .           | 360        |
| 9.27 tcComms.h File Reference . . . . .         | 360        |
| 9.27.1 Detailed Description . . . . .           | 361        |
| 9.28 TpyToEpics.cpp File Reference . . . . .    | 361        |
| 9.28.1 Detailed Description . . . . .           | 362        |
| 9.29 TpyToEpics.h File Reference . . . . .      | 362        |
| 9.29.1 Detailed Description . . . . .           | 363        |
| 9.30 TpyToEpicsConst.h File Reference . . . . . | 364        |
| 9.30.1 Detailed Description . . . . .           | 365        |
| <b>Index</b>                                    | <b>367</b> |



# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

|  |    |
|--|----|
| Device support for TwinCAT/ADS . . . . .       | 13 |
| Device driver functions . . . . .              | 22 |
| Info interface classes and functions . . . . . | 23 |
| TwinCAT tpy file parser . . . . .              | 25 |
| XML tpy file constants . . . . .               | 28 |
| Parser utility functions and classes . . . . . | 30 |
| OPC property constants . . . . .               | 33 |
| String functions and classes . . . . .         | 48 |
| TwinCAT read/write scanning . . . . .          | 56 |
| EPICS utility functions and classes . . . . .  | 57 |
| EPICS constants . . . . .                      | 61 |



# Chapter 2

## Namespace Index

### 2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

|                           |  |     |
|---------------------------|--|-----|
| <a href="#">DevTc</a>     | Namespace for TCat device support . . . . .        | 75  |
| <a href="#">EpicsTpy</a>  | Namespace for tpy-db conversion . . . . .          | 82  |
| <a href="#">InfoPlc</a>   | Namespace for Info communication . . . . .         | 84  |
| <a href="#">ParseTpy</a>  | Namespace for parsing . . . . .                    | 85  |
| <a href="#">ParseUtil</a> | Namespace for parsing utilities . . . . .          | 88  |
| <a href="#">plc</a>       | Namespace for abstract plc functionality . . . . . | 91  |
| <a href="#">TcComms</a>   | Namespace for TCat communication . . . . .         | 100 |



# Chapter 3

## Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

|   |     |
|---|-----|
| ParseTpy::ads_routing_info . . . . .            | 103 |
| ParseTpy::project_record . . . . .              | 275 |
| TcComms::AmsRouterNotification . . . . .        | 106 |
| std::atomic_string< stringT > . . . . .         | 110 |
| std::atomic_string< string > . . . . .          | 110 |
| std::atomic< string > . . . . .                 | 107 |
| std::atomic_string< wstring > . . . . .         | 110 |
| std::atomic< wstring > . . . . .                | 108 |
| ParseTpy::base_record . . . . .                 | 111 |
| ParseTpy::item_record . . . . .                 | 248 |
| ParseTpy::symbol_record . . . . .               | 286 |
| ParseTpy::type_record . . . . .                 | 322 |
| plc::BasePLC . . . . .                          | 114 |
| TcComms::TcPLC . . . . .                        | 299 |
| ParseUtil::bit_location . . . . .               | 146 |
| ParseTpy::item_record . . . . .                 | 248 |
| ParseTpy::type_record . . . . .                 | 322 |
| char_traits . . . . .                           |     |
| std::case_char_traits . . . . .                 | 147 |
| std::case_wchar_traits . . . . .                | 149 |
| ParseTpy::compiler_info . . . . .               | 150 |
| ParseTpy::project_record . . . . .              | 275 |
| TcComms::DataPar . . . . .                      | 151 |
| plc::DataValueTraits< T > . . . . .             | 185 |
| plc::DataValueTypeDef . . . . .                 | 187 |
| plc::BaseRecord . . . . .                       | 125 |
| plc::DataValue . . . . .                        | 152 |
| DevTc::devTcDeflo< RecType > . . . . .          | 191 |
| DevTc::devTcDefIn< RecType > . . . . .          | 189 |
| DevTc::devTcDefOut< RecType > . . . . .         | 193 |
| DevTc::devTcDefWaveformIn< RecType > . . . . .  | 194 |
| DevTc::epics_record_traits< RecType > . . . . . | 227 |
| std::std::hash< std::stringcase > . . . . .     | 236 |

|  |     |
|--|-----|
| std::std::hash< std::wstringcase > . . . . .                 | 237 |
| plc::Interface . . . . .                                     | 245 |
| DevTc::EpicsInterface . . . . .                              | 233 |
| InfoPlc::InfoInterface . . . . .                             | 238 |
| TcComms::TCatInterface . . . . .                             | 295 |
| EpicsTpy::macro_info . . . . .                               | 249 |
| EpicsTpy::macro_record . . . . .                             | 250 |
| ParseUtil::memory_location . . . . .                         | 251 |
| ParseTpy::symbol_record . . . . .                            | 286 |
| EpicsTpy::multi_io_support . . . . .                         | 255 |
| EpicsTpy::epics_macrofiles_processing . . . . .              | 220 |
| ParseUtil::opc_list . . . . .                                | 259 |
| ParseUtil::optarg . . . . .                                  | 260 |
| ParseTpy::parserinfo_type . . . . .                          | 263 |
| ParseUtil::process_arg . . . . .                             | 266 |
| InfoPlc::process_arg_info . . . . .                          | 270 |
| ParseUtil::process_arg_tc . . . . .                          | 272 |
| DevTc::register_devsup . . . . .                             | 276 |
| ParseUtil::replacement_rules . . . . .                       | 278 |
| EpicsTpy::epics_conversion . . . . .                         | 196 |
| EpicsTpy::epics_db_processing . . . . .                      | 201 |
| DevTc::epics_tc_db_processing . . . . .                      | 228 |
| EpicsTpy::epics_list_processing . . . . .                    | 213 |
| EpicsTpy::epics_macrofiles_processing . . . . .              | 220 |
| plc::scanner_thread_args . . . . .                           | 279 |
| EpicsTpy::split_io_support . . . . .                         | 280 |
| EpicsTpy::epics_db_processing . . . . .                      | 201 |
| EpicsTpy::epics_list_processing . . . . .                    | 213 |
| syminfo_processing . . . . .                                 | 287 |
| plc::System . . . . .  | 288 |
| ParseUtil::tag_processing . . . . .                          | 291 |
| ParseTpy::tpy_file . . . . .                                 | 311 |
| TcComms::tcProcWrite . . . . .                               | 307 |
| DevTc::tcRegisterTolocShell . . . . .                        | 310 |
| DevTc::epics_record_traits< RecType >::traits_type . . . . . | 320 |
| type_multipmap   |     |
| ParseTpy::type_map . . . . .                                 | 321 |
| ParseUtil::variable_name . . . . .                           | 324 |

# Chapter 4

## Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|                                |                                  |     |
|--------------------------------|----------------------------------|-----|
| ParseTpy::ads_routing_info     | ADS routing information          | 103 |
| TcComms::AmsRouterNotification | AMS Router Notification          | 106 |
| std::atomic< string >          | Atomic<string>                   | 107 |
| std::atomic< wstring >         | Atomic<wstring>                  | 108 |
| std::atomic_string< stringT >  | Atomic strings                   | 110 |
| ParseTpy::base_record          | Base record definition           | 111 |
| plc::BasePLC                   | Base PLC                         | 114 |
| plc::BaseRecord                | Class for managing a tag/channel | 125 |
| ParseUtil::bit_location        | Bit location                     | 146 |
| std::case_char_traits          | Case insensitive traits          | 147 |
| std::case_wchar_traits         | Case insensitive unicode traits  | 149 |
| ParseTpy::compiler_info        | Compiler information             | 150 |
| TcComms::DataPar               | Memory location struct           | 151 |
| plc::DataValue                 | Data value                       | 152 |
| plc::DataValueTraits< T >      | Data value traits                | 185 |
| plc::DataValueTypeDef          | Collection of type definitions   | 187 |
| DevTc::devTcDefIn< RecType >   | Device support input record      | 189 |
| DevTc::devTcDefIo< RecType >   | Device support record            | 191 |

|  |     |
|--|-----|
| DevTc::devTcDefOut< RecType >                      | 193 |
| Device support output record . . . . .             |     |
| DevTc::devTcDefWaveformIn< RecType >               | 194 |
| Device support waveform record . . . . .           |     |
| EpicsTyp::epics_conversion                         | 196 |
| Epics conversion . . . . .                         |     |
| EpicsTyp::epics_db_processing                      | 201 |
| Pics database record processing . . . . .          |     |
| EpicsTyp::epics_list_processing                    | 213 |
| List processing . . . . .                          |     |
| EpicsTyp::epics_macrofiles_processing              | 220 |
| Macro file processing . . . . .                    |     |
| DevTc::epics_record_traits< RecType >              | 227 |
| Epics record traits . . . . .                      |     |
| DevTc::epics_tc_db_processing                      | 228 |
| EPICS/TCat db processing . . . . .                 |     |
| DevTc::EpicsInterface                              | 233 |
| Epics interface class . . . . .                    |     |
| std::std::hash< std::stringcase >                  | 236 |
| Hash for case insensitive string . . . . .         |     |
| std::std::hash< std::wstringcase >                 | 237 |
| Hash for case insensitive unicode string . . . . . |     |
| InfoPlc::InfoInterface                             | 238 |
| Info interface . . . . .                           |     |
| plc::Interface                                     | 245 |
| Abstract interface . . . . .                       |     |
| ParseTyp::item_record                              | 248 |
| Item record . . . . .                              |     |
| EpicsTyp::macro_info                               | 249 |
| Macro information . . . . .                        |     |
| EpicsTyp::macro_record                             | 250 |
| Macro record . . . . .                             |     |
| ParseUtil::memory_location                         | 251 |
| Memory location . . . . .                          |     |
| EpicsTyp::multi_io_support                         | 255 |
| Multiple IO support . . . . .                      |     |
| ParseUtil::opc_list                                | 259 |
| OPC list . . . . .                                 |     |
| ParseUtil::optarg                                  | 260 |
| Optional arguments . . . . .                       |     |
| ParseTyp::parserinfo_type                          | 263 |
| Parser information . . . . .                       |     |
| ParseUtil::process_arg                             | 266 |
| Arguments for processing . . . . .                 |     |
| InfoPlc::process_arg_info                          | 270 |
| Arguments for processing . . . . .                 |     |
| ParseUtil::process_arg_tc                          | 272 |
| Arguments for processing . . . . .                 |     |
| ParseTyp::project_record                           | 275 |
| Project information . . . . .                      |     |
| DevTc::register_devsup                             | 276 |
| Device support registration . . . . .              |     |
| ParseUtil::replacement_rules                       | 278 |
| Replacement rules . . . . .                        |     |
| plc::scanner_thread_args                           | 279 |
| Scanner thread arguments . . . . .                 |     |
| EpicsTyp::split_io_support                         | 280 |
| Split IO support . . . . .                         |     |

|  |     |
|--|-----|
| <a href="#">ParseTpy::symbol_record</a>                                  |     |
| Symbol record  | 286 |
| <a href="#">syminfo_processing</a>                                       |     |
| Symbol processing  | 287 |
| <a href="#">plc::System</a>  |     |
| System to keep track of PLCs   | 288 |
| <a href="#">ParseUtil::tag_processing</a>                                |     |
| Tag processing selection   | 291 |
| <a href="#">TcComms::TCatInterface</a>                                   |     |
| TCat interface class   | 295 |
| <a href="#">TcComms::TcPLC</a>   |     |
| TwinCAT PLC  | 299 |
| <a href="#">TcComms::tcProcWrite</a>                                     |     |
| TwinCAT process write requests   | 307 |
| <a href="#">DevTc::tcRegisterTolocShell</a>                              |     |
| Register TC commands   | 310 |
| <a href="#">ParseTpy::tpy_file</a>                                       |     |
| Tpy file parsing   | 311 |
| <a href="#">DevTc::epics_record_traits&lt; RecType &gt;::traits_type</a> |     |
| Epics record type  | 320 |
| <a href="#">ParseTpy::type_map</a>                                       |     |
| Type dictionary  | 321 |
| <a href="#">ParseTpy::type_record</a>                                    |     |
| Type record information  | 322 |
| <a href="#">ParseUtil::variable_name</a>                                 |     |
| Variable name  | 324 |



# Chapter 5

## File Index

### 5.1 File List

Here is a list of all documented files with brief descriptions:

|                              |     |
|------------------------------|-----|
| atomic_string.h . . . . .    | 327 |
| devTc.cpp . . . . .          | 328 |
| devTc.h . . . . .            | 330 |
| drvTc.cpp . . . . .          | 332 |
| drvTc.h . . . . .            | 334 |
| EpicsDbGen.cpp . . . . .     | 335 |
| infoPlc.cpp . . . . .        | 336 |
| infoPlc.h . . . . .          | 337 |
| infoPlcTemplate.h . . . . .  | 338 |
| iocMain.cpp . . . . .        | 339 |
| ParseTpy.cpp . . . . .       | 340 |
| ParseTpy.h . . . . .         | 341 |
| ParseTpyConst.h . . . . .    | 343 |
| ParseTpyInfo.cpp . . . . .   | 345 |
| ParseTpyTemplate.h . . . . . | 347 |
| ParseUtil.cpp . . . . .      | 347 |
| ParseUtil.h . . . . .        | 348 |
| ParseUtilConst.h . . . . .   | 349 |
| plcBase.cpp . . . . .        | 351 |
| plcBase.h . . . . .          | 352 |
| plcBaseTemplate.h . . . . .  | 355 |
| stdafx.cpp . . . . .         | 356 |
| stdafx.h . . . . .           | 356 |
| stringcase.h . . . . .       | 357 |
| stringcase_hash.h . . . . .  | 358 |
| tcComms.cpp . . . . .        | 359 |
| tcComms.h . . . . .          | 360 |
| TpyToEpics.cpp . . . . .     | 361 |
| TpyToEpics.h . . . . .       | 362 |
| TpyToEpicsConst.h . . . . .  | 364 |



# Chapter 6

## Module Documentation

### 6.1 Device support for TwinCAT/ADS

#### Classes

- class `DevTc::EpicsInterface`  
*Epics interface class.*
- struct `DevTc::epics_record_traits< RecType >::traits_type`  
*Epics record type.*
- struct `DevTc::epics_record_traits< RecType >`  
*Epics record traits.*
- struct `DevTc::devTcDeflo< RecType >`  
*Device support record.*
- struct `DevTc::devTcDefIn< RecType >`  
*Device support input record.*
- struct `DevTc::devTcDefOut< RecType >`  
*device support output record.*
- struct `DevTc::devTcDefWaveformIn< RecType >`  
*device support waveform record.*

#### Typedefs

- typedef auto `DevTc::register_devsup::link_func(dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`  
-> bool  
*Type descriping the link function.*
- typedef std::pair< std::regex, link\_func & > `DevTc::register_devsup::test_pattern`  
*pair of pattern and link function*
- typedef std::vector< test\_pattern > `DevTc::register_devsup::test_pattern_list`  
*list of pattern/link functions*
- typedef epicsFloat64 `DevTc::epics_record_traits< RecType >::value_type`  
*Value type of (raw) value field.*
- typedef `epics_record_traits< RecType >::traits_type DevTc::devTcDeflo< RecType >::rec_type`  
*Record type: aiRecord, etc.*
- typedef `rec_type * DevTc::devTcDeflo< RecType >::rec_type_ptr`  
*Pointer to record type.*

## Enumerations

- enum `DevTc::epics_record_enum` {
 `DevTc::aaival` = 0, `DevTc::aaoval`, `DevTc::aival`, `DevTc::aoval`,
 `DevTc::bival`, `DevTc::boval`, `DevTc::eventval`, `DevTc::histogramval`,
 `DevTc::longinval`, `DevTc::longoutval`, `DevTc::mbbival`, `DevTc::mbboval`,
 `DevTc::mbbiDirectval`, `DevTc::mbboDirectval`, `DevTc::stringinval`, `DevTc::stringoutval`,
 `DevTc::waveformval`, `DevTc::airval`, `DevTc::aorval`, `DevTc::birval`,
 `DevTc::borval`, `DevTc::mbbiDirectrval`, `DevTc::mbboDirectrval`, `DevTc::mbbirval`,
 `DevTc::mbborval`, `DevTc::epics_record_enumEnd`, `DevTc::invalidval` = -1 }

*Epics record type enum.*

## Functions

- static void `DevTc::register_devsup::add` (const std::regex &rgx, `link_func` &func)  
*Register a pattern/link function.*
- static bool `DevTc::register_devsup::linkRecord` (const `std::stringcase` &inpout, dbCommon \*pEpicsRecord, plc::BaseRecordPtr &pRecord)  
*linkRecord*
- `DevTc::register_devsup::register_devsup ()`  
*Default constructor (adds linkTcRecord entry)*
- `DevTc::register_devsup::register_devsup (const register_devsup &)`  
*Disabled copy constructor.*
- `register_devsup & DevTc::register_devsup::operator= (const register_devsup &)`  
*Disabled assignment operator.*
- `DevTc::EpicsInterface::EpicsInterface (plc::BaseRecord &dval)`  
*Constructor.*
- `DevTc::EpicsInterface::~EpicsInterface ()`  
*Deconstructor.*
- void `DevTc::EpicsInterface::set_isPassive` (bool passive)  
*Set isPassive.*
- bool `DevTc::EpicsInterface::get_isCallback` () const  
*Get isCallback.*
- void `DevTc::EpicsInterface::set_isCallback` (bool isCb)  
*Set isCallback.*
- void `DevTc::EpicsInterface::set_pEpicsRecord` (dbCommon \*pEpRecord)  
*Set pEpicsRecord.*
- bool `DevTc::EpicsInterface::get_callbackRequestPending` () const  
*Get callbackRequestPending.*
- const CALLBACK & `DevTc::EpicsInterface::callback ()` const  
*Get pointer to callback structure.*
- CALLBACK & `DevTc::EpicsInterface::callback ()`  
*Get pointer to callback structure.*
- const IOSCANPVT & `DevTc::EpicsInterface::ioscan ()` const  
*Get reference to io scan list pointer.*
- IOSCANPVT & `DevTc::EpicsInterface::ioscan ()`  
*Get reference to io scan list pointer.*
- IOSCANPVT `DevTc::EpicsInterface::get_ioscan ()` const  
*Get pointer to io scan list.*
- void `DevTc::EpicsInterface::set_ioscan (const IOSCANPVT ioscan)`  
*Set pointer to io scan list.*

- virtual bool `DevTc::EpicsInterface::push ()` override  
*Makes a call to the EPICS dbProcess function.*
- virtual bool `DevTc::EpicsInterface::pull ()` override  
*Does nothing.*
- static int `DevTc::EpicsInterface::get_callback_queue_size (int pri)`
- static int `DevTc::EpicsInterface::get_callback_queue_used (int pri)`
- static int `DevTc::EpicsInterface::get_callback_queue_free (int pri)`
- void `DevTc::EpicsInterface::ioscan_reset (int bitnum)`  
*Reset ioscan use flag.*
- static const char \*const `DevTc::epics_record_traits< RecType >::name ()`  
*Name of the record.*
- static `value_type * DevTc::epics_record_traits< RecType >::val (traits_type *prec)`  
*Returns the (raw) value of a record.*
- static bool `DevTc::epics_record_traits< RecType >::read (traits_type *epicsrec, plc::BaseRecord *baserec)`  
*Performs the read access on prec.*
- static bool `DevTc::epics_record_traits< RecType >::write (plc::BaseRecord *baserec, traits_type *epicsrec)`  
*Performs the write access on prec.*
- `DevTc::devTcDeflo< RecType >::devTcDeflo ()`  
*Hide constructor.*
- static long `DevTc::devTcDeflo< RecType >::get_ioint_info (int cmd, dbCommon *prec, IOSCANPVT *ppvt)`  
*IO/INT info callback.*
- `DevTc::devTcDefIn< RecType >::devTcDefIn ()`  
*Constructor.*
- static long `DevTc::devTcDefIn< RecType >::init_read_record (rec_type_ptr prec)`  
*init callback for read records*
- static long `DevTc::devTcDefIn< RecType >::read (rec_type_ptr precord)`  
*read callback*
- `DevTc::devTcDefOut< RecType >::devTcDefOut ()`  
*Constructor.*
- static long `DevTc::devTcDefOut< RecType >::init_write_record (rec_type_ptr prec)`  
*init callback for write records*
- static long `DevTc::devTcDefOut< RecType >::write (rec_type_ptr precord)`  
*write callback*
- `DevTc::devTcDefWaveformIn< RecType >::devTcDefWaveformIn ()`  
*Constructor.*
- static long `DevTc::devTcDefWaveformIn< RecType >::init_read_waveform_record (rec_type_ptr prec)`  
*init callback for read records*
- static long `DevTc::devTcDefWaveformIn< RecType >::read_waveform (rec_type_ptr precord)`  
*read callback*

## Variables

- `test_pattern_list DevTc::register_devsup::tp_list`  
*list of pattern and links*
- static `register_devsup DevTc::register_devsup::the_register_devsup`  
*the one global instance of the register class*
- bool `DevTc::EpicsInterface::isPassive`
- bool `DevTc::EpicsInterface::isCallback`
- dbCommon \* `DevTc::EpicsInterface::pEpicsRecord`  
*Pointer to the EPICS record.*

- std::mutex [DevTc::EpicsInterface::ioscanmux](#)  
*IOSCAN mutex.*
- IOSCANPVT [DevTc::EpicsInterface::ioscanpvt](#)  
*Pointer to IO scan list.*
- std::atomic< unsigned int > [DevTc::EpicsInterface::ioscan\\_inuse](#)  
*Scan in progress (bit encoded value from priorities)*
- CALLBACK [DevTc::EpicsInterface::callbackval](#)  
*Callback structure.*
- double [DevTc::epics\\_record\\_traits< RecType >::traits\\_type::val](#)  
*Value.*
- static const aitEnum [DevTc::epics\\_record\\_traits< RecType >::value\\_ait\\_type](#) = aitEnumFloat64
- static const aitInt32 [DevTc::epics\\_record\\_traits< RecType >::value\\_count](#) = 0
- static const int [DevTc::epics\\_record\\_traits< RecType >::value\\_conversion](#) = 0  
*return value for read\_io functions 0=default, 2=don't convert*
- static const bool [DevTc::epics\\_record\\_traits< RecType >::input\\_record](#) = true  
*Indicates if this is an input record.*
- static const bool [DevTc::epics\\_record\\_traits< RecType >::raw\\_record](#) = false  
*Indicates if this is a raw record.*
- long [DevTc::devTcDeflo< RecType >::number](#)  
*Number of support functions.*
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::report\\_fn](#)  
*Report support function.*
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::init\\_fn](#)  
*Init support function.*
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::init\\_record\\_fn](#)  
*Record init support function.*
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::get\\_ioint\\_info\\_fn](#)  
*IO/INT support function.*
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::io\\_fn](#)  
*Read/write support function.*
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::special\\_linconv\\_fn](#)  
*Linear conversion support function.*

## Friends

- void [DevTc::EpicsInterface::complete\\_io\\_scan](#) ([EpicsInterface](#) \*, IOSCANPVT, int)

### 6.1.1 Detailed Description

### 6.1.2 Enumeration Type Documentation

#### 6.1.2.1 epics\_record\_enum

```
enum DevTc::epics\_record\_enum
```

Epics record type enum.

This record type enums are used as index the epics traits class

## Enumerator

|                      |                              |
|----------------------|------------------------------|
| aaival               | double input array           |
| aooval               | double output array          |
| aival                | double input                 |
| aoval                | double output                |
| bival                | binary input                 |
| boval                | binary output                |
| eventval             | event                        |
| histogramval         | histogram                    |
| longinval            | integer input                |
| longoutval           | integer output               |
| mbbival              | enum input                   |
| mbboval              | enum output                  |
| mbbiDirectval        | enum input direct            |
| mbboDirectval        | enum output direct           |
| stringinval          | string input                 |
| stringoutval         | string output                |
| waveformval          | waveform                     |
| airval               | raw double input             |
| aorval               | raw double output            |
| birval               | raw binary input             |
| borval               | raw binary output            |
| mbbiDirectrvl        | raw enum input               |
| mbboDirectrvl        | raw enum output              |
| mbbirval             | raw enum input direct        |
| mbborval             | raw enum output direct       |
| epics_record_enumEnd | End of enum (sentinel value) |
| invalidval           | invalid                      |

Definition at line 225 of file devTc.h.

### 6.1.3 Function Documentation

#### 6.1.3.1 `get_callback_queue_free()`

```
static int DevTc::EpicsInterface::get_callback_queue_free (
    int pri ) [static]
```

Get the free entries in the callback ring buffer For this function to return a valid value the EPICS distribution needs to be patched. Add the following lines:

```
epicsShareFunc epicsRingPointerId tcat_callbackQueue (int Priority)
{
    return (Priority >= 0) && (Priority < NUM_CALLBACK_PRIORITIES) ?
        callbackQueue[Priority].queue : NULL;
}
```

after the declaration of

```
static cbQueueSet callbackQueue[NUM_CALLBACK_PRIORITIES];
```

in src\ioc\db\callback.c

**Parameters**

|            |                         |
|------------|-------------------------|
| <i>pri</i> | Priority of ring buffer |
|------------|-------------------------|

**Returns**

free entries in the callback ring buffer

**6.1.3.2 get\_callback\_queue\_size()**

```
static int DevTc::EpicsInterface::get_callback_queue_size (
    int pri ) [static]
```

Get the size of the callback ring buffer For this function to return a valid value the EPICS distribution needs to be patched. Add the following lines:

```
epicsShareFunc epicsRingPointerId tcat_callbackQueue (int Priority)
{
    return (Priority >= 0) && (Priority < NUM_CALLBACK_PRIORITIES) ?
        callbackQueue[Priority].queue : NULL;
}
```

after the declaration of

```
static cbQueueSet callbackQueue[NUM_CALLBACK_PRIORITIES];
```

in src\ioc\db\callback.c

**Parameters**

|            |                         |
|------------|-------------------------|
| <i>pri</i> | Priority of ring buffer |
|------------|-------------------------|

**Returns**

size of the callback ring buffer

**6.1.3.3 get\_callback\_queue\_used()**

```
static int DevTc::EpicsInterface::get_callback_queue_used (
    int pri ) [static]
```

Get the used entries in the callback ring buffer For this function to return a valid value the EPICS distribution needs to be patched. Add the following lines:

```
epicsShareFunc epicsRingPointerId tcat_callbackQueue (int Priority)
{
    return (Priority >= 0) && (Priority < NUM_CALLBACK_PRIORITIES) ?
        callbackQueue[Priority].queue : NULL;
}
```

after the declaration of

```
static cbQueueSet callbackQueue[NUM_CALLBACK_PRIORITIES];
```

in src\ioc\db\callback.c

#### Parameters

|            |                         |
|------------|-------------------------|
| <i>pri</i> | Priority of ring buffer |
|------------|-------------------------|

#### Returns

used entries in the callback ring buffer

### 6.1.3.4 linkRecord()

```
bool DevTc::register_devsup::linkRecord (
    const std::string& inpout,
    dbCommon * pEpicsRecord,
    plc::BaseRecordPtr & pRecord ) [static]
```

linkRecord

Go through list and call first link function which matches the pattern Used to link epics records with internal records.

#### Parameters

|                     |                                   |
|---------------------|-----------------------------------|
| <i>inpout</i>       | Value of INP/OUT field            |
| <i>pEpicsRecord</i> | Pointer to EPICS record           |
| <i>pRecord</i>      | Pointer to a base record (return) |

#### Returns

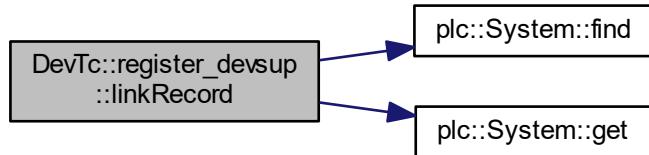
true if one match was found and successfully linked

[register\\_devsup::linkRecord](#)

Definition at line 112 of file devTc.cpp.

References plc::System::find(), and plc::System::get().

Here is the call graph for this function:



### 6.1.3.5 register\_devsup()

`DevTc::register_devsup::register_devsup ( ) [protected]`

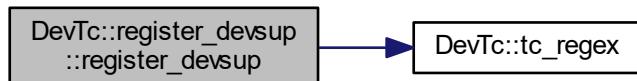
Default constructor (adds linkTcRecord entry)

[register\\_devsup::register\\_devsup](#)

Definition at line 104 of file devTc.cpp.

References `DevTc::tc_regex()`.

Here is the call graph for this function:



## 6.1.4 Variable Documentation

### 6.1.4.1 isCallback

`bool DevTc::EpicsInterface::isCallback [protected]`

Bool indicating whether callback is needed to call dbProcess true : SCAN = I/O Intr or the record is an out record

Definition at line 208 of file devTc.h.

Referenced by `DevTc::EpicsInterface::get_isCallback()`, `DevTc::EpicsInterface::push()`, and `DevTc::EpicsInterface::set_isCallback()`.

#### 6.1.4.2 **isPassive**

```
bool DevTc::EpicsInterface::isPassive [protected]
```

Bool indicating passive scan true : EPICS record SCAN field is set to PASSIVE

Definition at line 205 of file devTc.h.

Referenced by DevTc::EpicsInterface::push(), and DevTc::EpicsInterface::set\_isPassive().

#### 6.1.4.3 **the\_register\_devsup**

```
register_devsup DevTc::register_devsup::the_register_devsup [static], [protected]
```

the one global instance of the register class

**register\_devsup::the\_register\_devsup**

Definition at line 84 of file devTc.h.

Referenced by DevTc::register\_devsup::add().

#### 6.1.4.4 **value\_ait\_type**

```
template<epics_record_enum RecType>
const aitEnum DevTc::epics_record_traits< RecType >::value_ait_type = aitEnumFloat64 [static]
```

Data type of records val/rval field aitEnumInvalid type signals an array. Take type/len from record

Definition at line 300 of file devTc.h.

#### 6.1.4.5 **value\_count**

```
template<epics_record_enum RecType>
const aitInt32 DevTc::epics_record_traits< RecType >::value_count = 0 [static]
```

Array length: 1=scalar value, 0=array - see nelm for length, fixed for strings according to the record

Definition at line 303 of file devTc.h.

## 6.2 Device driver functions

### Namespaces

- [DevTc](#)

*Namespace for TCat device support.*

#### 6.2.1 Detailed Description

## 6.3 Info interface classes and functions

### Classes

- class `InfoPlc::InfoInterface`  
*Info interface.*
- class `InfoPlc::process_arg_info`  
*Arguments for processing.*

### Typedefs

- `typedef bool(InfoInterface::* InfoPlc::info_update_method) ()`  
*Pointer to info update method.*
- `typedef std::tuple< ParseUtil::variable_name, ParseUtil::process_type_enum, ParseUtil::opc_list, std::stringcase, bool, update_enum, info_update_method > InfoPlc::info_dbrecord_type`
- `typedef std::vector< info_dbrecord_type > InfoPlc::info_dbrecord_list`  
*List type of db info tuples.*

### Enumerations

- enum `InfoPlc::update_enum { InfoPlc::update_enum::forever, InfoPlc::update_enum::once, InfoPlc::update_enum::done }`  
*Update frequency type.*

#### 6.3.1 Detailed Description

#### 6.3.2 Typedef Documentation

##### 6.3.2.1 `info_dbrecord_type`

```
typedef std::tuple<ParseUtil::variable_name, ParseUtil::process_type_enum, ParseUtil::opc_list,
std::stringcase, bool, update_enum, info_update_method> InfoPlc::info_dbrecord_type
```

db info tuple with variable name, process type enum, opc list, TwinCAT type string, readonly, update enum, and update method

Definition at line 46 of file infoPlc.h.

#### 6.3.3 Enumeration Type Documentation

##### 6.3.3.1 `update_enum`

```
enum InfoPlc::update_enum [strong]
```

Update frequency type.

**Enumerator**

|         |                  |
|---------|------------------|
| forever | Repeat foreveer. |
| once    | once             |
| done    | done             |

Definition at line 27 of file infoPlc.h.

## 6.4 TwinCAT tpy file parser

### Classes

- class [ParseTpy::ads\\_routing\\_info](#)  
*ADS routing information.*
- class [ParseTpy::compiler\\_info](#)  
*Compiler information.*
- class [ParseTpy::project\\_record](#)  
*Project information.*
- class [ParseTpy::base\\_record](#)  
*Base record definition.*
- class [ParseTpy::item\\_record](#)  
*item record*
- class [ParseTpy::type\\_record](#)  
*Type record information.*
- class [ParseTpy::type\\_map](#)  
*Type dictionary.*
- class [ParseTpy::symbol\\_record](#)  
*Symbol record.*
- class [ParseTpy::tpy\\_file](#)  
*Tpy file parsing.*

### Typedefs

- typedef std::pair< int, int > [ParseTpy::dimension](#)
- typedef std::list< [dimension](#) > [ParseTpy::dimensions](#)
- typedef std::map< int, std::stringcase > [ParseTpy::enum\\_map](#)
- typedef std::pair< int, std::stringcase > [ParseTpy::enum\\_pair](#)
- typedef std::list< [item\\_record](#) > [ParseTpy::item\\_list](#)
- typedef std::multimap< unsigned int, [type\\_record](#) > [ParseTpy::type\\_multipmap](#)
- typedef std::list< [symbol\\_record](#) > [ParseTpy::symbol\\_list](#)

### Enumerations

- enum [ParseTpy::type\\_enum](#) {  
  [ParseTpy::type\\_enum::unknown](#), [ParseTpy::type\\_enum::simple](#), [ParseTpy::type\\_enum::arraytype](#), [ParseTpy::type\\_enum::enumtype](#),  
  [ParseTpy::type\\_enum::structtype](#), [ParseTpy::type\\_enum::functionblock](#) }  
*Type enum.*

#### 6.4.1 Detailed Description

#### 6.4.2 Typedef Documentation

#### 6.4.2.1 dimension

```
typedef std::pair<int, int> ParseTyp::dimension
```

This class stores a lbound, elements pair.

Definition at line 221 of file ParseTyp.h.

#### 6.4.2.2 dimensions

```
typedef std::list<dimension> ParseTyp::dimensions
```

This list stores lbound, elements pairs.

Definition at line 225 of file ParseTyp.h.

#### 6.4.2.3 enum\_map

```
typedef std::map<int, std::stringcase> ParseTyp::enum_map
```

This map stores a list of enum values.

Definition at line 229 of file ParseTyp.h.

#### 6.4.2.4 enum\_pair

```
typedef std::pair<int, std::stringcase> ParseTyp::enum_pair
```

This type stores an enum pair.

Definition at line 233 of file ParseTyp.h.

#### 6.4.2.5 item\_list

```
typedef std::list<item_record> ParseTyp::item_list
```

This class stores a list of subitems.

Definition at line 247 of file ParseTyp.h.

#### 6.4.2.6 symbol\_list

```
typedef std::list<symbol_record> ParseTpy::symbol_list
```

This is a list of symbol records

Definition at line 354 of file ParseTpy.h.

#### 6.4.2.7 type\_multimap

```
typedef std::multimap<unsigned int, type_record> ParseTpy::type_multimap
```

This is a multimap to store type records

Definition at line 316 of file ParseTpy.h.

### 6.4.3 Enumeration Type Documentation

#### 6.4.3.1 type\_enum

```
enum ParseTpy::type_enum [strong]
```

Type enum.

This structure describes a type record

Enumerator

|               |                  |
|---------------|------------------|
| unknown       | Unknown type.    |
| simple        | Simple type.     |
| arraytype     | Array type.      |
| enumtype      | Enumerated type. |
| structtype    | Structure type.  |
| functionblock | Function block.  |

Definition at line 253 of file ParseTpy.h.

## 6.5 XML tpy file constants

### Variables

- const char \*const ParseTpy::xmlPlcProjectInfo = "PlcProjectInfo"  
*PLC project info.*
- const char \*const ParseTpy::xmlProjectInfo = "ProjectInfo"  
*Project info.*
- const char \*const ParseTpy::xmlRoutingInfo = "RoutingInfo"  
*Routing info.*
- const char \*const ParseTpy::xmlCompilerInfo = "CompilerInfo"  
*Compiler info.*
- const char \*const ParseTpy::xmlAdsInfo = "AdsInfo"  
*ADS info.*
- const char \*const ParseTpy::xmlDataTypes = "DataTypes"  
*Data types.*
- const char \*const ParseTpy::xmlDataType = "DataType"  
*Data type.*
- const char \*const ParseTpy::xmlSymbols = "Symbols"  
*Symbols.*
- const char \*const ParseTpy::xmlSymbol = "Symbol"  
*Symbol.*
- const char \*const ParseTpy::xmlProperties = "Properties"  
*Properties.*
- const char \*const ParseTpy::xmlProperty = "Property"  
*Property.*
- const char \*const ParseTpy::xmlCompilerVersion = "CompilerVersion"  
*Compiler version.*
- const char \*const ParseTpy::xmlTwinCATVersion = "TwinCATVersion"  
*TwinCAT version.*
- const char \*const ParseTpy::xmlCpuFamily = "CpuFamily"  
*CPU family.*
- const char \*const ParseTpy::xmlNetId = "NetId"  
*Net ID.*
- const char \*const ParseTpy::xmlPort = "Port"  
*Port.*
- const char \*const ParseTpy::xmlTargetName = "TargetName"  
*Target name.*
- const char \*const ParseTpy::xmlName = "Name"  
*Name.*
- const char \*const ParseTpy::xmlType = "Type"  
*Type.*
- const char \*const ParseTpy::xmlAttrDecoration = "Decoration"  
*Decoration.*
- const char \*const ParseTpy::xmlAttrPointer = "Pointer"  
*Pointer.*
- const char \*const ParseTpy::xmlIGroup = "IGroup"  
*I Group.*
- const char \*const ParseTpy::xmlIOffset = "IOffset"  
*I Offset.*

- const char \*const ParseTpy::xmlBitSize = "BitSize"  
*Bit size.*
- const char \*const ParseTpy::xmlBitOffs = "BitOffs"  
*Bit Offset.*
- const char \*const ParseTpy::xmlArrayInfo = "ArrayInfo"  
*Array info.*
- const char \*const ParseTpy::xmlArrayLBound = "LBound"  
*Lower bound.*
- const char \*const ParseTpy::xmlArrayElements = "Elements"  
*Elements.*
- const char \*const ParseTpy::xmlSubItem = "SubItem"  
*Sub item.*
- const char \*const ParseTpy::xmlFbInfo = "FbInfo"  
*Fb info.*
- const char \*const ParseTpy::xmlEnumInfo = "EnumInfo"  
*Enum info.*
- const char \*const ParseTpy::xmlEnumText = "Text"  
*Text.*
- const char \*const ParseTpy::xmlEnumEnum = "Enum"  
*Enum.*
- const char \*const ParseTpy::xmlEnumComment = "Comment"  
*Comment.*
- const char \*const ParseTpy::xmlValue = "Value"  
*Value.*
- const char \*const ParseTpy::xmlDesc = "Desc"  
*Description.*
- const char \*const ParseTpy::opcExport = "opc"  
*OPC.*
- const char \*const ParseTpy::opcProp = "opc\_prop"  
*OPC property.*
- const char \*const ParseTpy::opcBracket = "["  
*OPC bracket.*

### 6.5.1 Detailed Description

## 6.6 Parser utility functions and classes

### Classes

- class `ParseUtil::replacement_rules`  
*Replacement rules.*
- class `ParseUtil::optarg`  
*Optional arguments.*
- class `ParseUtil::opc_list`  
*OPC list.*
- class `ParseUtil::variable_name`  
*Variable name.*
- class `ParseUtil::bit_location`  
*Bit location.*
- class `ParseUtil::memory_location`  
*Memory location.*
- class `ParseUtil::process_arg`  
*Arguments for processing.*
- class `ParseUtil::process_arg_tc`  
*Arguments for processing.*
- class `ParseUtil::tag_processing`  
*Tag processing selection.*

### Typedefs

- typedef `std::map< std::stringcase, std::stringcase > ParseUtil::replacement_table`
- typedef `std::map< int, std::stringcase > ParseUtil::property_map`
- typedef `std::pair< int, std::stringcase > ParseUtil::property_el`

### Enumerations

- enum `ParseUtil::opc_enum` { `ParseUtil::opc_enum::no_change`, `ParseUtil::opc_enum::publish`, `ParseUtil::opc_enum::silent` }  
*OPC state enum.*
- enum `ParseUtil::process_type_enum` {  
`ParseUtil::process_type_enum::pt_invalid`, `ParseUtil::process_type_enum::pt_int`, `ParseUtil::process_type_enum::pt_real`,  
`ParseUtil::process_type_enum::pt_bool`,  
`ParseUtil::process_type_enum::pt_string`, `ParseUtil::process_type_enum::pt_enum`, `ParseUtil::process_type_enum::pt_binary` }  
*Process type.*
- enum `ParseUtil::process_tag_enum` { `ParseUtil::process_tag_enum::all`, `ParseUtil::process_tag_enum::atomic`,  
`ParseUtil::process_tag_enum::structured` }  
*Tag preoicing enum.*

#### 6.6.1 Detailed Description

#### 6.6.2 Typedef Documentation

### 6.6.2.1 **property\_el**

```
typedef std::pair<int, std::stringcase> ParseUtil::property_el
```

This pair stores one element of opc properties.

Definition at line 139 of file ParseUtil.h.

### 6.6.2.2 **property\_map**

```
typedef std::map<int, std::stringcase> ParseUtil::property_map
```

This map stores a list of opc properties.

Definition at line 135 of file ParseUtil.h.

### 6.6.2.3 **replacement\_table**

```
typedef std::map<std::stringcase, std::stringcase> ParseUtil::replacement_table
```

Table of replacement rules

Definition at line 21 of file ParseUtil.h.

## 6.6.3 Enumeration Type Documentation

### 6.6.3.1 **opc\_enum**

```
enum ParseUtil::opc_enum [strong]
```

OPC state enum.

This enum denotes the opc state.

Enumerator

|           |                                    |
|-----------|------------------------------------|
| no_change | Do not change inherited behaviour. |
| publish   | Publish.                           |
| silent    | Do not publish.                    |

Definition at line 123 of file ParseUtil.h.

---

### 6.6.3.2 process\_tag\_enum

```
enum ParseUtil::process_tag_enum [strong]
```

Tag preoicessing enum.

Enumerated type to describe the tag processing

Enumerator

|            |  |
|------------|--|
| all        | Process all data types.                                      |
| atomic     | Process atomic data types.                                   |
| structured | Process structured data type (array, struct, function block) |

Definition at line 431 of file ParseUtil.h.

### 6.6.3.3 process\_type\_enum

```
enum ParseUtil::process_type_enum [strong]
```

Process type.

Enumerated type to describe the process type

Enumerator

|            |                      |
|------------|----------------------|
| pt_invalid | Invalid type.        |
| pt_int     | Numeral type.        |
| pt_real    | Floating point type. |
| pt_bool    | Logic type.          |
| pt_string  | String type.         |
| pt_enum    | Enumerated type.     |
| pt_binary  | Binary type.         |

Definition at line 322 of file ParseUtil.h.

## 6.7 OPC property constants

### Variables

- const int ParseUtil::OPC\_PROP\_CDT = 1
- const int ParseUtil::OPC\_PROP\_VALUE = 2
- const int ParseUtil::OPC\_PROP\_QUALITY = 3
- const int ParseUtil::OPC\_PROP\_TIME = 4
- const int ParseUtil::OPC\_PROP\_RIGHTS = 5
- const int ParseUtil::OPC\_PROP\_SCANRATE = 6
- const int ParseUtil::OPC\_PROP\_UNIT = 100
- const int ParseUtil::OPC\_PROP\_DESC = 101
- const int ParseUtil::OPC\_PROP\_HIEU = 102
- const int ParseUtil::OPC\_PROP\_LOEU = 103
- const int ParseUtil::OPC\_PROP\_HIRANGE = 104
- const int ParseUtil::OPC\_PROP\_LORANGE = 105
- const int ParseUtil::OPC\_PROP\_CLOSE = 106
- const int ParseUtil::OPC\_PROP\_OPEN = 107
- const int ParseUtil::OPC\_PROP\_TIMEZONE = 108
- const int ParseUtil::OPC\_PROP\_FGC = 201
- const int ParseUtil::OPC\_PROP\_BGC = 202
- const int ParseUtil::OPC\_PROP\_BLINK = 203
- const int ParseUtil::OPC\_PROP\_BMP = 204
- const int ParseUtil::OPC\_PROP SND = 205
- const int ParseUtil::OPC\_PROP\_HTML = 206
- const int ParseUtil::OPC\_PROP\_AVI = 207
- const int ParseUtil::OPC\_PROP\_ALMSTAT = 300
- const int ParseUtil::OPC\_PROP\_ALMHELP = 301
- const int ParseUtil::OPC\_PROP\_ALMAREAS = 302
- const int ParseUtil::OPC\_PROP\_ALMPRIMARYAREA = 303
- const int ParseUtil::OPC\_PROP\_ALMCCONDITION = 304
- const int ParseUtil::OPC\_PROP\_ALMLIMIT = 305
- const int ParseUtil::OPC\_PROP\_ALMDB = 306
- const int ParseUtil::OPC\_PROP\_ALMH = 307
- const int ParseUtil::OPC\_PROP\_ALMH = 308
- const int ParseUtil::OPC\_PROP\_ALML = 309
- const int ParseUtil::OPC\_PROP\_ALMLL = 310
- const int ParseUtil::OPC\_PROP\_ALMROC = 311
- const int ParseUtil::OPC\_PROP\_ALMDEV = 312
- const int ParseUtil::OPC\_PROP\_PREC = 8500
- const int ParseUtil::OPC\_PROP\_ZRST = 8510
- const int ParseUtil::OPC\_PROP\_FFST = 8525
- const int ParseUtil::OPC\_PROP\_RECTYPE = 8600
- const int ParseUtil::OPC\_PROP\_INOUT = 8601
- const char \*const ParseUtil::OPC\_PROP\_INPUT = "input"
- const char \*const ParseUtil::OPC\_PROP\_OUTPUT = "output"
- const int ParseUtil::OPC\_PROP\_TSE = 8602
- const int ParseUtil::OPC\_PROP\_PINI = 8603
- const int ParseUtil::OPC\_PROP\_DTYP = 8604
- const int ParseUtil::OPC\_PROP\_SERVER = 8610
- const int ParseUtil::OPC\_PROP\_PLNAME = 8611
- const int ParseUtil::OPC\_PROP\_ALIAS = 8620
- const int ParseUtil::OPC\_PROP\_ALMOSV = 8700
- const int ParseUtil::OPC\_PROP\_ALMZSV = 8701

- const int ParseUtil::OPC\_PROP\_ALMCOSV = 8702
- const int ParseUtil::OPC\_PROP\_ALMUNSV = 8703
- const int ParseUtil::OPC\_PROP\_ALMZRSV = 8710
- const int ParseUtil::OPC\_PROP\_ALMFFSV = 8725
- const int ParseUtil::OPC\_PROP\_ALMHHSV = 8727
- const int ParseUtil::OPC\_PROP\_ALMHHSV = 8728
- const int ParseUtil::OPC\_PROP\_ALMLSV = 8729
- const int ParseUtil::OPC\_PROP\_ALMLLSV = 8730
- const int ParseUtil::OPC\_PROP\_FIELD\_BEG = 8800
- const int ParseUtil::OPC\_PROP\_FIELD\_END = 9000

### 6.7.1 Detailed Description

### 6.7.2 Variable Documentation

#### 6.7.2.1 OPC\_PROP\_ALIAS

const int ParseUtil::OPC\_PROP\_ALIAS = 8620

alias for structure item or symbol name

Definition at line 69 of file ParseUtilConst.h.

Referenced by ParseUtil::variable\_name::append(), and EpicsTpy::epics\_db\_processing::operator()().

#### 6.7.2.2 OPC\_PROP\_ALMAREAS

const int ParseUtil::OPC\_PROP\_ALMAREAS = 302

area

Definition at line 45 of file ParseUtilConst.h.

#### 6.7.2.3 OPC\_PROP\_ALMCONDITION

const int ParseUtil::OPC\_PROP\_ALMCONDITION = 304

condition

Definition at line 47 of file ParseUtilConst.h.

#### 6.7.2.4 OPC\_PROP\_ALMCOSV

```
const int ParseUtil::OPC_PROP_ALMCOSV = 8702
```

alarm: change of state severity

Definition at line 72 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

#### 6.7.2.5 OPC\_PROP\_ALMDB

```
const int ParseUtil::OPC_PROP_ALMDB = 306
```

dead band, tolerance

Definition at line 49 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

#### 6.7.2.6 OPC\_PROP\_ALMDEV

```
const int ParseUtil::OPC_PROP_ALMDEV = 312
```

deviation

Definition at line 55 of file ParseUtilConst.h.

#### 6.7.2.7 OPC\_PROP\_ALMFFSV

```
const int ParseUtil::OPC_PROP_ALMFFSV = 8725
```

alarm: fifteen state severity

Definition at line 75 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.8 OPC\_PROP\_ALMH

```
const int ParseUtil::OPC_PROP_ALMH = 308
```

high alarm -> HIGH

Definition at line 51 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.9 OPC\_PROP\_ALMHELP

```
const int ParseUtil::OPC_PROP_ALMHELP = 301
```

help

Definition at line 44 of file ParseUtilConst.h.

### 6.7.2.10 OPC\_PROP\_ALMHH

```
const int ParseUtil::OPC_PROP_ALMHH = 307
```

high high alarm -> HIHI

Definition at line 50 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.11 OPC\_PROP\_ALMHHSV

```
const int ParseUtil::OPC_PROP_ALMHHSV = 8727
```

alarm: hihi severity

Definition at line 76 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.12 OPC\_PROP\_ALMHSV

```
const int ParseUtil::OPC_PROP_ALMHSV = 8728
```

alarm: high severity

Definition at line 77 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.13 OPC\_PROP\_ALML

```
const int ParseUtil::OPC_PROP_ALML = 309
```

low alarm -> LOW

Definition at line 52 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.14 OPC\_PROP\_ALMLIMIT

```
const int ParseUtil::OPC_PROP_ALMLIMIT = 305
```

limit

Definition at line 48 of file ParseUtilConst.h.

### 6.7.2.15 OPC\_PROP\_ALMLL

```
const int ParseUtil::OPC_PROP_ALMLL = 310
```

low low alaam -> LOLO

Definition at line 53 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.16 OPC\_PROP\_ALMLLSV

const int ParseUtil::OPC\_PROP\_ALMLLSV = 8730

alarm: lolo severity

Definition at line 79 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.17 OPC\_PROP\_ALMLSV

const int ParseUtil::OPC\_PROP\_ALMLSV = 8729

alarm: low severity

Definition at line 78 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.18 OPC\_PROP\_ALMOSV

const int ParseUtil::OPC\_PROP\_ALMOSV = 8700

alarm: one severity

Definition at line 70 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.19 OPC\_PROP\_ALMPRIMARYAREA

const int ParseUtil::OPC\_PROP\_ALMPRIMARYAREA = 303

primery area

Definition at line 46 of file ParseUtilConst.h.

### 6.7.2.20 OPC\_PROP\_ALMROC

const int ParseUtil::OPC\_PROP\_ALMROC = 311

rate of change

Definition at line 54 of file ParseUtilConst.h.

### 6.7.2.21 OPC\_PROP\_ALMSTAT

```
const int ParseUtil::OPC_PROP_ALMSTAT = 300  
status
```

Definition at line 43 of file ParseUtilConst.h.

### 6.7.2.22 OPC\_PROP\_ALMUNSV

```
const int ParseUtil::OPC_PROP_ALMUNSV = 8703  
alarm: unknown state severity
```

Definition at line 73 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.23 OPC\_PROP\_ALMZRSV

```
const int ParseUtil::OPC_PROP_ALMZRSV = 8710  
alarm: one state severity
```

Definition at line 74 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.24 OPC\_PROP\_ALMZSV

```
const int ParseUtil::OPC_PROP_ALMZSV = 8701  
alarm: zero severity
```

Definition at line 71 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.25 OPC\_PROP\_AVI

```
const int ParseUtil::OPC_PROP_AVI = 207  
avi file
```

Definition at line 41 of file ParseUtilConst.h.

### 6.7.2.26 OPC\_PROP\_BGC

const int ParseUtil::OPC\_PROP\_BGC = 202

background color

Definition at line 36 of file ParseUtilConst.h.

### 6.7.2.27 OPC\_PROP\_BLINK

const int ParseUtil::OPC\_PROP\_BLINK = 203

blinking

Definition at line 37 of file ParseUtilConst.h.

### 6.7.2.28 OPC\_PROP\_BMP

const int ParseUtil::OPC\_PROP\_BMP = 204

bmp file

Definition at line 38 of file ParseUtilConst.h.

### 6.7.2.29 OPC\_PROP\_CDT

const int ParseUtil::OPC\_PROP\_CDT = 1

canonocal data type

Definition at line 18 of file ParseUtilConst.h.

### 6.7.2.30 OPC\_PROP\_CLOSE

const int ParseUtil::OPC\_PROP\_CLOSE = 106

label for close state -> ONAM

Definition at line 31 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()(), and EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.31 OPC\_PROP\_DESC

```
const int ParseUtil::OPC_PROP_DESC = 101
```

description string -> DESC

Definition at line 26 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.32 OPC\_PROP\_DTYP

```
const int ParseUtil::OPC_PROP_DTYP = 8604
```

DTYP field: opc or opcRaw

Definition at line 66 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.33 OPC\_PROP\_FFST

```
const int ParseUtil::OPC_PROP_FFST = 8525
```

... fifteen string

Definition at line 59 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()(), and EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.34 OPC\_PROP\_FGC

```
const int ParseUtil::OPC_PROP_FGC = 201
```

foreground color

Definition at line 35 of file ParseUtilConst.h.

### 6.7.2.35 OPC\_PROP\_FIELD\_BEG

```
const int ParseUtil::OPC_PROP_FIELD_BEG = 8800
```

Beginning of field,value combination

Definition at line 80 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.36 OPC\_PROP\_FIELD\_END

```
const int ParseUtil::OPC_PROP_FIELD_END = 9000
```

End of field,value combinations

Definition at line 81 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.37 OPC\_PROP\_HIEU

```
const int ParseUtil::OPC_PROP_HIEU = 102
```

high expectation value -> HOPR

Definition at line 27 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.38 OPC\_PROP\_HIRANGE

```
const int ParseUtil::OPC_PROP_HIRANGE = 104
```

absolute maximum value -> DRVH

Definition at line 29 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.39 OPC\_PROP\_HTML

```
const int ParseUtil::OPC_PROP_HTML = 206
```

html file

Definition at line 40 of file ParseUtilConst.h.

### 6.7.2.40 OPC\_PROP\_INOUT

```
const int ParseUtil::OPC_PROP_INOUT = 8601
```

input or output

Definition at line 61 of file ParseUtilConst.h.

Referenced by ParseUtil::opc\_list::is\_READONLY(), and EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.41 OPC\_PROP\_INPUT

```
const char* const ParseUtil::OPC_PROP_INPUT = "input"
```

input

Definition at line 62 of file ParseUtilConst.h.

Referenced by ParseUtil::opc\_list::is\_READONLY().

### 6.7.2.42 OPC\_PROP\_LOEU

```
const int ParseUtil::OPC_PROP_LOEU = 103
```

low expectation value -> LOPR

Definition at line 28 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.43 OPC\_PROP\_LORANGE

const int ParseUtil::OPC\_PROP\_LORANGE = 105

absolute minimum value -> DRVL

Definition at line 30 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.44 OPC\_PROP\_OPEN

const int ParseUtil::OPC\_PROP\_OPEN = 107

label for open state -> ZNAM

Definition at line 32 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()(), and EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.45 OPC\_PROP\_OUTPUT

const char\* const ParseUtil::OPC\_PROP\_OUTPUT = "output"

output

Definition at line 63 of file ParseUtilConst.h.

Referenced by ParseUtil::opc\_list::is\_READONLY().

### 6.7.2.46 OPC\_PROP\_PINI

const int ParseUtil::OPC\_PROP\_PINI = 8603

initialization

Definition at line 65 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

**6.7.2.47 OPC\_PROP\_PLCNAME**

```
const int ParseUtil::OPC_PROP_PLCNAME = 8611
```

tc name including ads routing info and port

Definition at line 68 of file ParseUtilConst.h.

Referenced by ParseUtil::process\_arg\_tc::get\_full(), EpicsTpy::epics\_db\_processing::operator()(), and ParseTpy::tpy\_file::parse\_finish().

**6.7.2.48 OPC\_PROP\_PREC**

```
const int ParseUtil::OPC_PROP_PREC = 8500
```

precision

Definition at line 57 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

**6.7.2.49 OPC\_PROP\_QUALITY**

```
const int ParseUtil::OPC_PROP_QUALITY = 3
```

data quality flag

Definition at line 20 of file ParseUtilConst.h.

**6.7.2.50 OPC\_PROP\_RECTYPE**

```
const int ParseUtil::OPC_PROP_RECTYPE = 8600
```

record type

Definition at line 60 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.51 OPC\_PROP\_RIGHTS

const int ParseUtil::OPC\_PROP\_RIGHTS = 5

access right: 1 read, 2 write, 3 read/write

Definition at line 22 of file ParseUtilConst.h.

Referenced by ParseUtil::opc\_list::is\_READONLY().

### 6.7.2.52 OPC\_PROP\_SCANRATE

const int ParseUtil::OPC\_PROP\_SCANRATE = 6

scan rate

Definition at line 23 of file ParseUtilConst.h.

### 6.7.2.53 OPC\_PROP\_SERVER

const int ParseUtil::OPC\_PROP\_SERVER = 8610

server name

Definition at line 67 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.54 OPC\_PROP SND

const int ParseUtil::OPC\_PROP SND = 205

sound file

Definition at line 39 of file ParseUtilConst.h.

### 6.7.2.55 OPC\_PROP\_TIME

const int ParseUtil::OPC\_PROP\_TIME = 4

timestamp

Definition at line 21 of file ParseUtilConst.h.

### 6.7.2.56 OPC\_PROP\_TIMEZONE

const int ParseUtil::OPC\_PROP\_TIMEZONE = 108

time zone

Definition at line 33 of file ParseUtilConst.h.

### 6.7.2.57 OPC\_PROP\_TSE

const int ParseUtil::OPC\_PROP\_TSE = 8602

time stamp

Definition at line 64 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.58 OPC\_PROP\_UNIT

const int ParseUtil::OPC\_PROP\_UNIT = 100

unit string -> EGU

Definition at line 25 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()(), and EpicsTpy::epics\_db\_processing::operator()().

### 6.7.2.59 OPC\_PROP\_VALUE

const int ParseUtil::OPC\_PROP\_VALUE = 2

value

Definition at line 19 of file ParseUtilConst.h.

### 6.7.2.60 OPC\_PROP\_ZRST

const int ParseUtil::OPC\_PROP\_ZRST = 8510

zero string ...

Definition at line 58 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()(), and EpicsTpy::epics\_db\_processing::operator()().

---

## 6.8 String functions and classes

### Classes

- struct `std::case_char_traits`  
*case insensitive traits.*
- struct `std::case_wchar_traits`  
*case insensitive unicode traits.*

### Typedefs

- `typedef std::basic_string< char, case_char_traits > std::stringcase`  
*case insensitive string.*
- `typedef std::basic_string< wchar_t, case_wchar_traits > std::wstringcase`  
*case insensitive string.*

### Functions

- int `std::strncasecmp` (const char \*s1, const char \*s2, size\_t n)  
*case insensitive compare with maximum length*
- int `std::wcscasewcmp` (const wchar\_t \*s1, const wchar\_t \*s2, size\_t n)  
*case insensitive unicode compare with maximum length*
- static bool `std::case_char_traits::eq` (const char\_type &c1, const char\_type &c2)
- static bool `std::case_char_traits::ne` (const char\_type &c1, const char\_type &c2)
- static bool `std::case_char_traits::lt` (const char\_type &c1, const char\_type &c2)
- static int `std::case_char_traits::compare` (const char\_type \*s1, const char\_type \*s2, size\_t n)
- static bool `std::case_wchar_traits::eq` (const char\_type &c1, const char\_type &c2)
- static bool `std::case_wchar_traits::ne` (const char\_type &c1, const char\_type &c2)
- static bool `std::case_wchar_traits::lt` (const char\_type &c1, const char\_type &c2)
- static int `std::case_wchar_traits::compare` (const char\_type \*s1, const char\_type \*s2, size\_t n)
- void `std::trim_space` (`std::stringcase &s`)
- void `std::trim_space` (`std::wstringcase &s`)
- template<class Container , class String , class Predicate >  
`void std::split_string` (Container &output, const String &input, const Predicate &pred, bool trimEmpty=true)  
*Splits a strings.*

#### 6.8.1 Detailed Description

#### 6.8.2 Typedef Documentation

##### 6.8.2.1 `stringcase`

```
typedef std::basic_string<char, case_char_traits> std::stringcase
case insensitive string.
```

This string class is not case sensitive.

Definition at line 104 of file stringcase.h.

### 6.8.2.2 wstringcase

```
typedef std::basic_string<wchar_t, case_wchar_traits> std::wstringcase
```

case insensitive string.

This string class is not case sensitive.

Definition at line 109 of file stringcase.h.

## 6.8.3 Function Documentation

### 6.8.3.1 compare() [1/2]

```
static int std::case_char_traits::compare (
    const char_type * s1,
    const char_type * s2,
    size_t n ) [inline], [static]
```

Compare strings

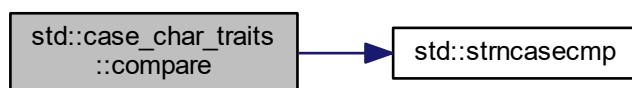
Parameters

|    |                      |
|----|----------------------|
| s1 | First string         |
| s2 | Second string        |
| n  | number of characters |

Definition at line 64 of file stringcase.h.

References std::strncasecmp().

Here is the call graph for this function:



### 6.8.3.2 compare() [2/2]

```
static int std::case_wchar_traits::compare (
    const char_type * s1,
    const char_type * s2,
    size_t n ) [inline], [static]
```

Compare strings

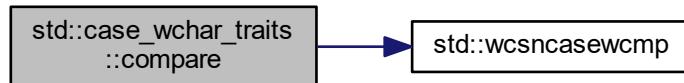
#### Parameters

|           |                      |
|-----------|----------------------|
| <i>s1</i> | First string         |
| <i>s2</i> | Second string        |
| <i>n</i>  | number of characters |

Definition at line 96 of file stringcase.h.

References std::wcsncasewcmp().

Here is the call graph for this function:



### 6.8.3.3 eq() [1/2]

```
static bool std::case_char_traits::eq (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Equal character

#### Parameters

|           |             |
|-----------|-------------|
| <i>c1</i> | First char  |
| <i>c2</i> | Second char |

Definition at line 45 of file stringcase.h.

**6.8.3.4 eq() [2/2]**

```
static bool std::case_wchar_traits::eq (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Equal character

**Parameters**

|           |             |
|-----------|-------------|
| <i>c1</i> | First char  |
| <i>c2</i> | Second char |

Definition at line 77 of file stringcase.h.

**6.8.3.5 lt() [1/2]**

```
static bool std::case_char_traits::lt (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Lower than character

**Parameters**

|           |             |
|-----------|-------------|
| <i>c1</i> | First char  |
| <i>c2</i> | Second char |

Definition at line 57 of file stringcase.h.

**6.8.3.6 lt() [2/2]**

```
static bool std::case_wchar_traits::lt (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Lower than character

**Parameters**

|           |             |
|-----------|-------------|
| <i>c1</i> | First char  |
| <i>c2</i> | Second char |

Definition at line 89 of file stringcase.h.

### 6.8.3.7 `ne()` [1/2]

```
static bool std::case_char_traits::ne (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Not equal character

#### Parameters

|           |             |
|-----------|-------------|
| <i>c1</i> | First char  |
| <i>c2</i> | Second char |

Definition at line 51 of file stringcase.h.

### 6.8.3.8 `ne()` [2/2]

```
static bool std::case_wchar_traits::ne (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Not equal character

#### Parameters

|           |             |
|-----------|-------------|
| <i>c1</i> | First char  |
| <i>c2</i> | Second char |

Definition at line 83 of file stringcase.h.

### 6.8.3.9 `split_string()`

```
template<class Container , class String , class Predicate >
void std::split_string (
    Container & output,
    const String & input,
    const Predicate & pred,
    bool trimEmpty = true )
```

Splits a strings.

Splits a string into its tokens and adds them to a container. The delimiter can be easily specified with a lambda expression. Example: `stringcase arg ("This is a test!"); vector<stringcase> list; split_string (list, arg, [] (char c)->bool { return isspace (c) != 0; }, true);`

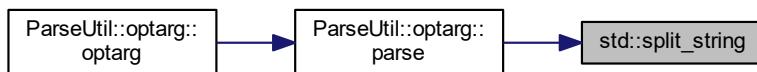
## Parameters

|                  |   |
|------------------|---|
| <i>output</i>    | Output container  |
| <i>input</i>     | Input string  |
| <i>pred</i>      | Function which returns true when character is a separator |
| <i>trimEmpty</i> | Trims empty strings when true                             |

Definition at line 135 of file stringcase.h.

Referenced by ParseUtil::optarg::parse().

Here is the caller graph for this function:



### 6.8.3.10 strncasecmp()

```
int std::strncasecmp (
    const char * s1,
    const char * s2,
    size_t n ) [inline]
```

case insensitive compare with maximum length

Case insensitive compare.

## Parameters

|           |                      |
|-----------|----------------------|
| <i>s1</i> | first string         |
| <i>s2</i> | second string        |
| <i>n</i>  | Number of characters |

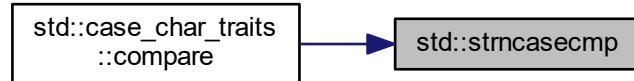
## Returns

<0 smaller, 0 equal, >0 greater

Definition at line 23 of file stringcase.h.

Referenced by std::case\_char\_traits::compare().

Here is the caller graph for this function:



### 6.8.3.11 trim\_space() [1/2]

```
void std::trim_space (
    std::stringcase & s )
```

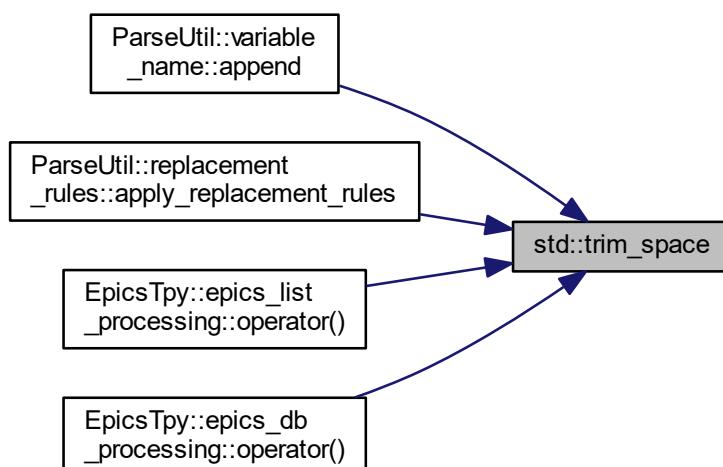
trim space on both ends.

#### Parameters

|   |                |
|---|----------------|
| s | string to trim |
|---|----------------|

Referenced by ParseUtil::variable\_name::append(), ParseUtil::replacement\_rules::apply\_replacement\_rules(), EpicsTpy::epics\_list\_processing::operator()(), and EpicsTpy::epics\_db\_processing::operator()().

Here is the caller graph for this function:



### 6.8.3.12 trim\_space() [2/2]

```
void std::trim_space (
    std::wstringcase & s )
```

trim space on both ends.

Parameters

|   |                |
|---|----------------|
| s | string to trim |
|---|----------------|

### 6.8.3.13 wcsncasewcmp()

```
int std::wcsncasewcmp (
    const wchar_t * s1,
    const wchar_t * s2,
    size_t n ) [inline]
```

case insensitive unicode compare with maximum length

Case insensitive compare for unicode string.

Parameters

|    |                      |
|----|----------------------|
| s1 | first string         |
| s2 | second string        |
| n  | Number of characters |

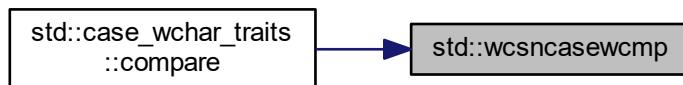
Returns

<0 smaller, 0 equal, >0 greater

Definition at line 34 of file stringcase.h.

Referenced by std::case\_wchar\_traits::compare().

Here is the caller graph for this function:



## 6.9 TwinCAT read/write scanning

### Classes

- struct **TcComms::DataPar**  
*Memory location struct.*
- class **TcComms::TCatInterface**  
*TCat interface class.*
- class **TcComms::tcProcWrite**  
*TwinCAT process write requests.*
- class **TcComms::TcPLC**  
*TwinCAT PLC.*
- class **TcComms::AmsRouterNotification**  
*AMS Router Notification.*

### Variables

- const int **TcComms::MAX\_REQ\_SIZE** = 250000  
*maximum allowed request size (bytes)*
- const int **TcComms::MAX\_SINGLE\_GAP\_SIZE** = 50  
*maximum allowed size (bytes) of a memory gap within continuous request*
- const double **TcComms::MAX\_REL\_GAP** = 0.25  
*(maximum allowed total gap size) / (current request size)*
- const int **TcComms::MIN\_REL\_GAP\_SIZE** = 100  
*minimum allowed relative gap size (bytes)*
- const int **TcComms::default\_scanrate** = 100  
*default PLC TwinCAT scan rate (100ms)*
- const int **TcComms::minimum\_scanrate** = 5  
*minimum PLC TwinCAT scan rate (5ms)*
- const int **TcComms::maximum\_scanrate** = 10000  
*maximum PLC TwinCAT scan rate (10s)*
- const int **TcComms::default\_multiple** = 10  
*default multiple for PLC EPICS scan rate (10)*
- const int **TcComms::minimum\_multiple** = 1  
*minimum multiple for PLC EPICS scan rate (1)*
- const int **TcComms::maximum\_multiple** = 200  
*maximum multiple for PLC EPICS scan rate (200)*

### 6.9.1 Detailed Description

## 6.10 EPICS utility functions and classes

### Classes

- class [EpicsTpy::epics\\_conversion](#)  
*Epics conversion.*
- class [EpicsTpy::split\\_io\\_support](#)  
*Split IO support.*
- class [EpicsTpy::multi\\_io\\_support](#)  
*Multiple IO support.*
- class [EpicsTpy::epics\\_list\\_processing](#)  
*List processing.*
- struct [EpicsTpy::macro\\_info](#)  
*Macro information.*
- struct [EpicsTpy::macro\\_record](#)  
*Macro record.*
- class [EpicsTpy::epics\\_macrofiles\\_processing](#)  
*Macro file processing.*
- class [EpicsTpy::epics\\_db\\_processing](#)  
*epics database record processing*

### Typedefs

- typedef std::vector< [macro\\_info](#) > [EpicsTpy::macro\\_list](#)
- typedef std::stack< [macro\\_record](#) > [EpicsTpy::macro\\_stack](#)
- typedef std::unordered\_set< std::stringcase > [EpicsTpy::filename\\_set](#)

### Enumerations

- enum [EpicsTpy::tc\\_epics\\_conv](#) { [EpicsTpy::tc\\_epics\\_conv::no\\_conversion](#), [EpicsTpy::tc\\_epics\\_conv::no\\_dot](#), [EpicsTpy::tc\\_epics\\_conv::ligo\\_std](#), [EpicsTpy::tc\\_epics\\_conv::ligo\\_vac](#) }  
*Conversion rules for TC/EPICS.*
- enum [EpicsTpy::case\\_type](#) { [EpicsTpy::case\\_type::preserve](#), [EpicsTpy::case\\_type::upper](#), [EpicsTpy::case\\_type::lower](#) }  
*Case conversion rule enum.*
- enum [EpicsTpy::io\\_filestat](#) { [EpicsTpy::io\\_filestat::closed](#), [read](#), [write](#) }  
*enum for file io*
- enum [EpicsTpy::listing\\_type](#) { [EpicsTpy::listing\\_type::standard](#), [EpicsTpy::listing\\_type::autoburt](#), [EpicsTpy::listing\\_type::daqini](#) }  
*Listing type enum.*
- enum [EpicsTpy::macrofile\\_type](#) { [EpicsTpy::macrofile\\_type::all](#), [EpicsTpy::macrofile\\_type::fields](#), [EpicsTpy::macrofile\\_type::error](#) }
- enum [EpicsTpy::device\\_support\\_type](#) { [EpicsTpy::device\\_support\\_type::opc\\_name](#), [EpicsTpy::device\\_support\\_type::tc\\_name](#) }  
*Device support enum.*

#### 6.10.1 Detailed Description

#### 6.10.2 Typedef Documentation

### 6.10.2.1 filename\_set

```
typedef std::unordered_set<std::stringcase> EpicsTpy::filename_set
```

A set of filenames

Definition at line 567 of file TpyToEpics.h.

### 6.10.2.2 macro\_list

```
typedef std::vector<macro_info> EpicsTpy::macro_list
```

A list of fields

Definition at line 540 of file TpyToEpics.h.

### 6.10.2.3 macro\_stack

```
typedef std::stack<macro_record> EpicsTpy::macro_stack
```

A stack of records/structs

Definition at line 563 of file TpyToEpics.h.

## 6.10.3 Enumeration Type Documentation

### 6.10.3.1 case\_type

```
enum EpicsTpy::case_type [strong]
```

Case conversion rule enum.

This enum describes the case conversion rule

Enumerator

|          |                        |
|----------|------------------------|
| preserve | Preserve the case.     |
| upper    | Convert to upper case. |
| lower    | Convert to lower case. |

Definition at line 46 of file TpyToEpics.h.

### 6.10.3.2 device\_support\_type

```
enum EpicsTpy::device_support_type [strong]
```

Device support enum.

This enum describes the type of listing to produce

Enumerator

|          |   |
|----------|---|
| opc_name | Use opc names in the INPUT/OUTPUT epics fields.     |
| tc_name  | Use TwinCAT names in the INPUT/OUTPUT epics fields. |

Definition at line 692 of file TpyToEpics.h.

### 6.10.3.3 io\_filestat

```
enum EpicsTpy::io_filestat [strong]
```

enum for file io

Enumerator

|        |                |
|--------|----------------|
| closed | file is closed |
|--------|----------------|

Definition at line 308 of file TpyToEpics.h.

### 6.10.3.4 listing\_type

```
enum EpicsTpy::listing_type [strong]
```

Listing type enum.

This enum describes the type of listing to produce

Enumerator

|          |   |
|----------|---|
| standard | Standard listing using TwinCAT/OPC names. |
| autoburt | Autoburt listing.                         |
| daqini   | LIGO DAQ ini listing.                     |

Definition at line 423 of file TpyToEpics.h.

### 6.10.3.5 macrofile\_type

```
enum EpicsTpy::macrofile_type [strong]
```

This enum describes the type of macros to produce

#### Enumerator

|        |  |
|--------|--|
| all    | Include all fields and error messages. |
| fields | Include all fields.                    |
| errors | Include error messages.                |

Definition at line 512 of file TpyToEpics.h.

### 6.10.3.6 tc\_epics\_conv

```
enum EpicsTpy::tc_epics_conv [strong]
```

Conversion rules for TC/EPICS.

This enum describes the TwinCAT/opc to EPICS conversion rule

#### Enumerator

|               |  |
|---------------|--|
| no_conversion | No conversion.   |
| no_dot        | Convert '.' to '_'.  |
| ligo_std      | LIGO standard conversion: Eliminate leading '.', Replace sceond '.' with ':', Replace third '.' with '-', Replace all other '.' with '_'                     |
| ligo_vac      | LIGO standard conversion for vacuum channels: Eliminate leading '.', Replace sceond '.' with ':', Replace third '.' with '_', Replace all other '.' with '_' |

Definition at line 24 of file TpyToEpics.h.

## 6.11 EPICS constants

### Variables

- const int `EpicsTpy::MAX_EPICS_CHANNEL` = 54
- const int `EpicsTpy::MAX_EPICS_DESC` = 40
- const int `EpicsTpy::MAX_EPICS_STRING` = 40
- const int `EpicsTpy::MAX_EPICS_UNIT` = 15
- const char \*const `EpicsTpy::EPICS_DB_EGU` = "EGU"
- const char \*const `EpicsTpy::EPICS_DB_DESC` = "DESC"
- const char \*const `EpicsTpy::EPICS_DB_HOPR` = "HOPR"
- const char \*const `EpicsTpy::EPICS_DB_LOPR` = "LOPR"
- const char \*const `EpicsTpy::EPICS_DB_DRVH` = "DRVH"
- const char \*const `EpicsTpy::EPICS_DB_DRVL` = "DRVL"
- const char \*const `EpicsTpy::EPICS_DB_ONAM` = "ONAM"
- const char \*const `EpicsTpy::EPICS_DB_ZNAM` = "ZNAM"
- const char \*const `EpicsTpy::EPICS_DB_PREC` = "PREC"
- const char \*const `EpicsTpy::EPICS_DB_ZRST` [16]
- const char \*const `EpicsTpy::EPICS_DB_ZRVL` [16]
- const char \*const `EpicsTpy::EPICS_DB_SCAN` = "SCAN"
- const char \*const `EpicsTpy::EPICS_DB_INP` = "INP"
- const char \*const `EpicsTpy::EPICS_DB_OUT` = "OUT"
- const char \*const `EpicsTpy::EPICS_DB_TSE` = "TSE"
- const char \*const `EpicsTpy::EPICS_DB_PINI` = "PINI"
- const char \*const `EpicsTpy::EPICS_DB_DTYP` = "DTYP"
- const char \*const `EpicsTpy::EPICS_DB_OSV` = "OSV"
- const char \*const `EpicsTpy::EPICS_DB_ZSV` = "ZSV"
- const char \*const `EpicsTpy::EPICS_DB_COSV` = "COSV"
- const char \*const `EpicsTpy::EPICS_DB_HIHI` = "HIHI"
- const char \*const `EpicsTpy::EPICS_DB_HIGH` = "HIGH"
- const char \*const `EpicsTpy::EPICS_DB_LOW` = "LOW"
- const char \*const `EpicsTpy::EPICS_DB_LOLO` = "LOLO"
- const char \*const `EpicsTpy::EPICS_DB_HYST` = "HYST"
- const char \*const `EpicsTpy::EPICS_DB_HHSV` = "HHSV"
- const char \*const `EpicsTpy::EPICS_DB_HSV` = "HSV"
- const char \*const `EpicsTpy::EPICS_DB_LSV` = "LSV"
- const char \*const `EpicsTpy::EPICS_DB_LLSV` = "LLSV"
- const char \*const `EpicsTpy::EPICS_DB_NOALARM` = "NO\_ALARM"
- const char \*const `EpicsTpy::EPICS_DB_MINOR` = "MINOR"
- const char \*const `EpicsTpy::EPICS_DB_MAJOR` = "MAJOR"
- const char \*const `EpicsTpy::EPICS_DB_UNSV` = "UNSV"
- const char \*const `EpicsTpy::EPICS_DB_ZRSV` [16]
- const char \*const `EpicsTpy::EPICS_DB_FORBIDDEN` []
- const char \*const `EpicsTpy::EPICS_DB_ALLOWED` []
- const char \*const `EpicsTpy::EPICS_DB_NUMVAL` []
- const char \*const `EpicsTpy::LIGODAQ_DATATYPE_NAME` = "datatype"
- const int `EpicsTpy::LIGODAQ_DATATYPE_FLOAT` = 4
- const int `EpicsTpy::LIGODAQ_DATATYPE_INT32` = 2
- const int `EpicsTpy::LIGODAQ_DATATYPE_DEFAULT` = `LIGODAQ_DATATYPE_FLOAT`
- const char \*const `EpicsTpy::LIGODAQ_UNIT_NAME` = "units"
- const char \*const `EpicsTpy::LIGODAQ_UNIT_NONE` = "none"
- const char \*const `EpicsTpy::LIGODAQ_UNIT_DEFAULT` = `LIGODAQ_UNIT_NONE`
- const char \*const `EpicsTpy::LIGODAQ_INI_HEADER`

### 6.11.1 Detailed Description

### 6.11.2 Variable Documentation

#### 6.11.2.1 EPICS\_DB\_ALLOWED

```
const char* const EpicsTpy::EPICS_DB_ALLOWED[ ]
```

**Initial value:**

```
= {"LINR", "EGUF", "EGUL", "AOFF", "ASLO", "ESLO", "SMOO", "HIHI", "LOLO", "HIGH", "LOW", "HHSV", "LLSV",
     "HSV", "LSV", "HYST", "ZSV", "OSV", "COSV", "IVOA", "IVOV", "UNSV", "ZRSV",
     "ONSV", "TWSV", "THSV",
     "FRSV", "FVSV", "SXSV", "SVSV", "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV",
     "FTSV", "FFSV", NULL}
```

Names of allowed EPICS record fields

Definition at line 75 of file TpyToEpicsConst.h.

#### 6.11.2.2 EPICS\_DB\_COSV

```
const char* const EpicsTpy::EPICS_DB_COSV = "COSV"
```

change severity

Definition at line 47 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

#### 6.11.2.3 EPICS\_DB\_DESC

```
const char* const EpicsTpy::EPICS_DB_DESC = "DESC"
```

description string

Definition at line 24 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

#### 6.11.2.4 EPICS\_DB\_DRVH

```
const char* const EpicsTpy::EPICS_DB_DRVH = "DRVH"
```

drive high string

Definition at line 27 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

#### 6.11.2.5 EPICS\_DB\_DRVL

```
const char* const EpicsTpy::EPICS_DB_DRVL = "DRVL"
```

drive low string

Definition at line 28 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

#### 6.11.2.6 EPICS\_DB\_DTYP

```
const char* const EpicsTpy::EPICS_DB_DTYP = "DTYP"
```

data type

Definition at line 43 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

#### 6.11.2.7 EPICS\_DB\_EGU

```
const char* const EpicsTpy::EPICS_DB_EGU = "EGU"
```

unit string

Definition at line 23 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.8 EPICS\_DB\_FORBIDDEN

```
const char* const EpicsTpy::EPICS_DB_FORBIDDEN[ ]
```

**Initial value:**

```
= { "", EPICS_DB_SCAN, "DOL", EPICS_DB_INP, EPICS_DB_OUT, "VAL", "RVAL", "INIT",
      "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
      "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of forbidden EPICS record fields

Definition at line 69 of file TpyToEpicsConst.h.

### 6.11.2.9 EPICS\_DB\_HHSV

```
const char* const EpicsTpy::EPICS_DB_HHSV = "HHSV"
```

high severity high limit

Definition at line 54 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.10 EPICS\_DB\_HIGH

```
const char* const EpicsTpy::EPICS_DB_HIGH = "HIGH"
```

low severity high limit

Definition at line 50 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.11 EPICS\_DB\_HIHI

```
const char* const EpicsTpy::EPICS_DB_HIHI = "HIHI"
```

high severity high limit

Definition at line 49 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.12 EPICS\_DB\_HOPR

```
const char* const EpicsTpy::EPICS_DB_HOPR = "HOPR"
```

high ops value string

Definition at line 25 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.13 EPICS\_DB\_HSV

```
const char* const EpicsTpy::EPICS_DB_HSV = "HSV"
```

low severity high limit

Definition at line 55 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.14 EPICS\_DB\_HYST

```
const char* const EpicsTpy::EPICS_DB_HYST = "HYST"
```

deadband

Definition at line 53 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.15 EPICS\_DB\_INP

```
const char* const EpicsTpy::EPICS_DB_INP = "INP"
```

input link

Definition at line 39 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.16 EPICS\_DB\_LLSV

```
const char* const EpicsTpy::EPICS_DB_LLSV = "LLSV"
```

high severity low limit

Definition at line 57 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.17 EPICS\_DB\_LOLO

```
const char* const EpicsTpy::EPICS_DB_LOLO = "LOLO"
```

high severity low limit

Definition at line 52 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.18 EPICS\_DB\_LOPR

```
const char* const EpicsTpy::EPICS_DB_LOPR = "LOPR"
```

low ops value string

Definition at line 26 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.19 EPICS\_DB\_LOW

```
const char* const EpicsTpy::EPICS_DB_LOW = "LOW"
```

low severity low limit

Definition at line 51 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.20 EPICS\_DB\_LSV

```
const char* const EpicsTpy::EPICS_DB_LSV = "LSV"
```

low severity low limit

Definition at line 56 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.21 EPICS\_DB\_MAJOR

```
const char* const EpicsTpy::EPICS_DB_MAJOR = "MAJOR"
```

major alarm

Definition at line 61 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()(), and EpicsTpy::epics\_db\_processing::process\_field←\_alarm().

### 6.11.2.22 EPICS\_DB\_MINOR

```
const char* const EpicsTpy::EPICS_DB_MINOR = "MINOR"
```

minor alarm

Definition at line 60 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()(), and EpicsTpy::epics\_db\_processing::process\_field←\_alarm().

### 6.11.2.23 EPICS\_DB\_NOALARM

```
const char* const EpicsTpy::EPICS_DB_NOALARM = "NO_ALARM"
```

no alarm

Definition at line 59 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::process\_field\_alarm().

### 6.11.2.24 EPICS\_DB\_NUMVAL

const char\* const EpicsTpy::EPICS\_DB\_NUMVAL[ ]

**Initial value:**

```
= {EPICS_DB_TSE, EPICS_DB_HOPR, EPICS_DB_Lopr, EPICS_DB_DrvH, EPICS_DB_DrvL, "EGUF", "EGUL",
    "AOFF", "ASLO", "ESLO", "SMOO", "PREC", "HIHI", "LOLO", "HIGH", "LOW", "HYST",
    "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL", "EIVL", "NIVL",
    "TEVL", "ELVL",
    "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of EPICS record fields which are numeric

Definition at line 81 of file TpyToEpicsConst.h.

### 6.11.2.25 EPICS\_DB\_ONAM

const char\* const EpicsTpy::EPICS\_DB\_ONAM = "ONAM"

one name string

Definition at line 29 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.26 EPICS\_DB\_OSV

const char\* const EpicsTpy::EPICS\_DB\_OSV = "OSV"

one severity

Definition at line 45 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.27 EPICS\_DB\_OUT

const char\* const EpicsTpy::EPICS\_DB\_OUT = "OUT"

output link

Definition at line 40 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

**6.11.2.28 EPICS\_DB\_PINI**

```
const char* const EpicsTpy::EPICS_DB_PINI = "PINI"
```

initialization

Definition at line 42 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

**6.11.2.29 EPICS\_DB\_PREC**

```
const char* const EpicsTpy::EPICS_DB_PREC = "PREC"
```

precision

Definition at line 32 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

**6.11.2.30 EPICS\_DB\_SCAN**

```
const char* const EpicsTpy::EPICS_DB_SCAN = "SCAN"
```

Scan

Definition at line 38 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

**6.11.2.31 EPICS\_DB\_TSE**

```
const char* const EpicsTpy::EPICS_DB_TSE = "TSE"
```

time stamp

Definition at line 41 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.32 EPICS\_DB\_UNSV

```
const char* const EpicsTpy::EPICS_DB_UNSV = "UNSV"
```

unknown severity

Definition at line 63 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.33 EPICS\_DB\_ZNAM

```
const char* const EpicsTpy::EPICS_DB_ZNAM = "ZNAM"
```

zero name string

Definition at line 30 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.34 EPICS\_DB\_ZRST

```
const char* const EpicsTpy::EPICS_DB_ZRST[16]
```

**Initial value:**

```
= {"ZRST", "ONST", "TWST", "THST", "FRST", "FVST", "SXST", "SVST",
    "EIST", "NIST", "TEST", "ELST", "TVST", "TTST", "FTST", "FFST"}
```

enum string

Definition at line 33 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.35 EPICS\_DB\_ZRSV

```
const char* const EpicsTpy::EPICS_DB_ZRSV[16]
```

**Initial value:**

```
= {"ZRSV", "ONSV", "TWSV", "THSV", "FRSV", "FVSV", "SXSV", "SVSV",
    "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV", "FTSV", "FFSV"}
```

enum alarm severity

Definition at line 64 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.36 EPICS\_DB\_ZRVL

```
const char* const EpicsTpy::EPICS_DB_ZRVL[16]
```

**Initial value:**

```
= {"ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
    "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL"}
```

enum val

Definition at line 35 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.37 EPICS\_DB\_ZSV

```
const char* const EpicsTpy::EPICS_DB_ZSV = "ZSV"
```

zero severity

Definition at line 46 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.38 LIGODAQ\_DATATYPE\_DEFAULT

```
const int EpicsTpy::LIGODAQ_DATATYPE_DEFAULT = LIGODAQ_DATATYPE_FLOAT
```

default DAQ datatype is float

Definition at line 91 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()().

### 6.11.2.39 LIGODAQ\_DATATYPE\_FLOAT

```
const int EpicsTpy::LIGODAQ_DATATYPE_FLOAT = 4
```

DAQ datatype is float

Definition at line 89 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()().

### 6.11.2.40 LIGODAQ\_DATATYPE\_INT32

```
const int EpicsTpy::LIGODAQ_DATATYPE_INT32 = 2
```

DAQ datatype is int32

Definition at line 90 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()().

### 6.11.2.41 LIGODAQ\_DATATYPE\_NAME

```
const char* const EpicsTpy::LIGODAQ_DATATYPE_NAME = "datatype"
```

DAQ datatype name

Definition at line 88 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()().

### 6.11.2.42 LIGODAQ\_INI\_HEADER

```
const char* const EpicsTpy::LIGODAQ_INI_HEADER
```

#### Initial value:

```
=
"[default]\n"
"gain=1.00\n"
"datatype=%i\n"
"ifoid=0\n"
"slope=6.1028e-05\n"
"acquire=3\n"
"offset=0\n"
"units=%s\n"
"dcuid=4\n"
"datarate=16"
```

DAQ ini file header: substitute defaults

Definition at line 98 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()().

### 6.11.2.43 LIGODAQ\_UNIT\_DEFAULT

```
const char* const EpicsTpy::LIGODAQ_UNIT_DEFAULT = LIGODAQ_UNIT_NONE
```

default DAQ unit is none

Definition at line 95 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()().

### 6.11.2.44 LIGODAQ\_UNIT\_NAME

```
const char* const EpicsTpy::LIGODAQ_UNIT_NAME = "units"
```

DAQ unit name

Definition at line 93 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()().

### 6.11.2.45 LIGODAQ\_UNIT\_NONE

```
const char* const EpicsTpy::LIGODAQ_UNIT_NONE = "none"
```

DAQ unit for no unit

Definition at line 94 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_list\_processing::operator()().

### 6.11.2.46 MAX\_EPICS\_CHANNEL

```
const int EpicsTpy::MAX_EPICS_CHANNEL = 54
```

maximum length of EPICS channel name

Definition at line 18 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.47 MAX\_EPICS\_DESC

```
const int EpicsTpy::MAX_EPICS_DESC = 40
```

maximum length of EPICS channel description

Definition at line 19 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics\_db\_processing::operator()().

### 6.11.2.48 MAX\_EPICS\_STRING

```
const int EpicsTpy::MAX_EPICS_STRING = 40
```

maximum length of EPICS strings

Definition at line 20 of file TpyToEpicsConst.h.

### 6.11.2.49 MAX\_EPICS\_UNIT

```
const int EpicsTpy::MAX_EPICS_UNIT = 15
```

maximum length of EPICS channel unit string

Definition at line 21 of file TpyToEpicsConst.h.



# Chapter 7

## Namespace Documentation

### 7.1 DevTc Namespace Reference

Namespace for TCat device support.

#### Classes

- struct `devTcDefIn`  
*Device support input record.*
- struct `devTcDefLo`  
*Device support record.*
- struct `devTcDefOut`  
*device support output record.*
- struct `devTcDefWaveformIn`  
*device support waveform record.*
- struct `epics_record_traits`  
*Epics record traits.*
- class `epics_tc_db_processing`  
*EPICS/TCat db processing.*
- class `EpicsInterface`  
*Epics interface class.*
- class `register_devsup`  
*Device support registration.*
- class `tcRegisterTolocShell`  
*Register TC commands.*

#### Typedefs

- typedef std::tuple< `std::stringcase`, `std::stringcase`, `epics_list_processing` \*, `bool` > `filename_rule_list_tuple`  
*Tuple for filnemae, rule and list processing.*
- typedef std::vector< `filename_rule_list_tuple` > `tc_listing_def`  
*List of tuples for filnemae, rule and list processing.*
- typedef std::tuple< `std::stringcase`, `std::stringcase`, `epics_macrofiles_processing` \*, `const char` \* > `dirname_arg_macro_tuple`  
*Tuple for directory name, argument and macro list processing.*
- typedef std::vector< `dirname_arg_macro_tuple` > `tc_macro_def`  
*List of tuples for directory name, argument and macro list processing.*

## Enumerations

- enum `epics_record_enum` {
 aaival = 0, `aaoval`, `aival`, `aoval`,
 `bival`, `boval`, `eventval`, `histogramval`,
 `longinval`, `longoutval`, `mbbival`, `mbboval`,
 `mbbiDirectval`, `mbboDirectval`, `stringinval`, `stringoutval`,
 `waveformval`, `airval`, `aorval`, `birval`,
 `borval`, `mbbiDirectrval`, `mbboDirectrval`, `mbbirval`,
 `mbborval`, `epics_record_enumEnd`, `invalidval` = -1 }

*Epics record type enum.*

## Functions

- const std::regex `tc_regex` ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.|.?) +:(8[0-9][0-9]))/(\\d{1,9})/(\\d{1,9})← :\\d{1,9}"))
 

*Callback for output record.*
- const std::regex `info_regex` ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.|.?) +:(8[0-9][0-9]))/(info)/([A-Za-z0-9\_]+)")
 

*Regex for indentifying info records.*
- void `tcLoadRecords` (const iocshArgBuf \*args)
 

*Load TwinCAT records.*
- void `tcSetScanRate` (const iocshArgBuf \*args)
 

*Set the scan rate.*
- void `tcList` (const iocshArgBuf \*args)
 

*Generate channel lists.*
- void `tcMacro` (const iocshArgBuf \*args)
 

*Generated macro files.*
- void `tcAlias` (const iocshArgBuf \*args)
 

*Define alias and replacement rules.*
- void `tcInfoPrefix` (const iocshArgBuf \*args)
 

*Sets the info prefix.*
- void `tcPrintVals` (const iocshArgBuf \*args)
 

*Print all values.*
- void `tcPrintVal` (const iocshArgBuf \*args)
 

*Print value.*

### 7.1.1 Detailed Description

Namespace for TCat device support.

`DevTc` Name space

`DevTc` namespace

### 7.1.2 Function Documentation

### 7.1.2.1 tc\_regex()

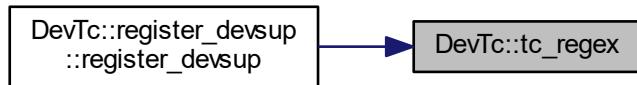
```
const std::regex DevTc::tc_regex (
    "((tc)://((\\b([0-9]|1[0-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?) +(8[0-9][0-9])) /) (\\d{1,9}) /"
    "(\\d{1,9})")
```

Callback for output record.

Callback function to process EPICS out record Regex for indentifying TwinCAT records

Referenced by DevTc::register\_devsup::register\_devsup().

Here is the caller graph for this function:



### 7.1.2.2 tcAlias()

```
void DevTc::tcAlias (
    const iocshArgBuf * args )
```

Define alias and replacement rules.

Define a nick name or alias

#### Parameters

|             |                       |
|-------------|-----------------------|
| <i>args</i> | Arguments for tcAlias |
|-------------|-----------------------|

Definition at line 760 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



### 7.1.2.3 tcInfoPrefix()

```
void DevTc::tcInfoPrefix (
    const iocshArgBuf * args )
```

Sets the info prefix.

Sets the channel prefix for info PLC records

#### Parameters

|      |                            |
|------|----------------------------|
| args | Arguments for tcInfoPrefix |
|------|----------------------------|

Definition at line 814 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



### 7.1.2.4 tcList()

```
void DevTc::tcList (
    const iocshArgBuf * args )
```

Generate channel lists.

List function to generate separate listings

**Parameters**

|             |                      |
|-------------|----------------------|
| <i>args</i> | Arguments for tcList |
|-------------|----------------------|

Definition at line 710 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



### 7.1.2.5 tcLoadRecords()

```
void DevTc::tcLoadRecords (
    const iocshArgBuf * args )
```

Load TwinCAT records.

Function for loading a TCat tpy file, and using it to generate internal record entries as well as the EPICs .db file

**Parameters**

|             |                             |
|-------------|-----------------------------|
| <i>args</i> | Arguments for tcLoadRecords |
|-------------|-----------------------------|

Definition at line 481 of file drvTc.cpp.

### 7.1.2.6 tcMacro()

```
void DevTc::tcMacro (
    const iocshArgBuf * args )
```

Generated macro files.

Macro function to generate macro files

**Parameters**

|             |                       |
|-------------|-----------------------|
| <i>args</i> | Arguments for tcMacro |
|-------------|-----------------------|

Definition at line 735 of file drvTc.cpp.

References `plc::System::get()`.

Here is the call graph for this function:

**7.1.2.7 tcPrintVal()**

```
void DevTc::tcPrintVal (const iocshArgBuf * args )
```

Print value.

Debugging function that prints the values for one or multiple records of the PLCs. Supports wildcards.

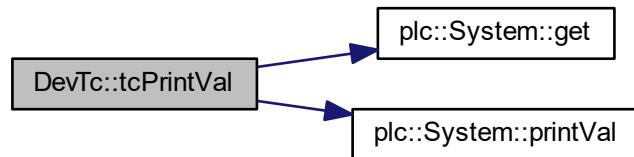
**Parameters**

|             |                          |
|-------------|--------------------------|
| <i>args</i> | Arguments for tcPrintVal |
|-------------|--------------------------|

Definition at line 856 of file drvTc.cpp.

References `plc::System::get()`, and `plc::System::printVal()`.

Here is the call graph for this function:



### 7.1.2.8 tcPrintVals()

```
void DevTc::tcPrintVals (
    const iocshArgBuf * args )
```

Print all values.

Debugging function that prints the values for all records of the PLCs

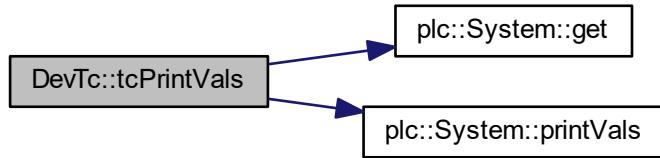
#### Parameters

|      |                           |
|------|---------------------------|
| args | Arguments for tcPrintVals |
|------|---------------------------|

Definition at line 845 of file drvTc.cpp.

References plc::System::get(), and plc::System::printVals().

Here is the call graph for this function:



### 7.1.2.9 tcSetScanRate()

```
void DevTc::tcSetScanRate (
    const iocshArgBuf * args )
```

Set the scan rate.

Set scan rate of the read scanner

#### Parameters

|      |                             |
|------|-----------------------------|
| args | Arguments for tcSetScanRate |
|------|-----------------------------|

Definition at line 653 of file drvTc.cpp.

References `plc::System::get()`.

Here is the call graph for this function:



## 7.2 EpicsTpy Namespace Reference

Namespace for tpy-db conversion.

### Classes

- class [epics\\_conversion](#)  
*Epics conversion.*
- class [epics\\_db\\_processing](#)  
*Epics database record processing*
- class [epics\\_list\\_processing](#)  
*List processing.*
- class [epics\\_macrofiles\\_processing](#)  
*Macro file processing.*
- struct [macro\\_info](#)  
*Macro information.*
- struct [macro\\_record](#)  
*Macro record.*
- class [multi\\_io\\_support](#)  
*Multiple IO support.*
- class [split\\_io\\_support](#)  
*Split IO support.*

### Typedefs

- typedef `std::vector<macro_info>` [macro\\_list](#)
- typedef `std::stack<macro_record>` [macro\\_stack](#)
- typedef `std::unordered_set<std::stringcase>` [filename\\_set](#)

## Enumerations

- enum `tc_epics_conv` { `tc_epics_conv::no_conversion`, `tc_epics_conv::no_dot`, `tc_epics_conv::ligo_std`, `tc_epics_conv::ligo_vac` }
 

*Conversion rules for TC/EPICS.*
- enum `case_type` { `case_type::preserve`, `case_type::upper`, `case_type::lower` }
 

*Case conversion rule enum.*
- enum `io_filestat` { `io_filestat::closed`, `read`, `write` }
 

*enum for file io*
- enum `listing_type` { `listing_type::standard`, `listing_type::autoburt`, `listing_type::daqini` }
 

*Listing type enum.*
- enum `macrofile_type` { `macrofile_type::all`, `macrofile_type::fields`, `macrofile_type::errors` }
- enum `device_support_type` { `device_support_type::opc_name`, `device_support_type::tc_name` }
 

*Device support enum.*

## Variables

- const int `MAX_EPICS_CHANNEL` = 54
- const int `MAX_EPICS_DESC` = 40
- const int `MAX_EPICS_STRING` = 40
- const int `MAX_EPICS_UNIT` = 15
- const char \*const `EPICS_DB_EGU` = "EGU"
- const char \*const `EPICS_DB_DESC` = "DESC"
- const char \*const `EPICS_DB_HOPR` = "HOPR"
- const char \*const `EPICS_DB_LOPR` = "LOPR"
- const char \*const `EPICS_DB_DRVH` = "DRVH"
- const char \*const `EPICS_DB_DRVL` = "DRVL"
- const char \*const `EPICS_DB_ONAM` = "ONAM"
- const char \*const `EPICS_DB_ZNAM` = "ZNAME"
- const char \*const `EPICS_DB_PREC` = "PREC"
- const char \*const `EPICS_DB_ZRST` [16]
- const char \*const `EPICS_DB_ZRVL` [16]
- const char \*const `EPICS_DB_SCAN` = "SCAN"
- const char \*const `EPICS_DB_INP` = "INP"
- const char \*const `EPICS_DB_OUT` = "OUT"
- const char \*const `EPICS_DB_TSE` = "TSE"
- const char \*const `EPICS_DB_PINI` = "PINI"
- const char \*const `EPICS_DB_DTYP` = "DTYP"
- const char \*const `EPICS_DB_OSV` = "OSV"
- const char \*const `EPICS_DB_ZSV` = "ZSV"
- const char \*const `EPICS_DB_COSV` = "COSV"
- const char \*const `EPICS_DB_HIHI` = "HIHI"
- const char \*const `EPICS_DB_HIGH` = "HIGH"
- const char \*const `EPICS_DB_LOW` = "LOW"
- const char \*const `EPICS_DB_LOLO` = "LOLO"
- const char \*const `EPICS_DB_HYST` = "HYST"
- const char \*const `EPICS_DB_HHSV` = "HHSV"
- const char \*const `EPICS_DB_HSV` = "HSV"
- const char \*const `EPICS_DB_LSV` = "LSV"
- const char \*const `EPICS_DB_LLSV` = "LLSV"
- const char \*const `EPICS_DB_NOALARM` = "NO\_ALARM"
- const char \*const `EPICS_DB_MINOR` = "MINOR"
- const char \*const `EPICS_DB_MAJOR` = "MAJOR"

- const char \*const EPICS\_DB\_UNSV = "UNSV"
- const char \*const EPICS\_DB\_ZRSV [16]
- const char \*const EPICS\_DB\_FORBIDDEN []
- const char \*const EPICS\_DB\_ALLOWED []
- const char \*const EPICS\_DB\_NUMVAL []
- const char \*const LIGODAQ\_DATATYPE\_NAME = "datatype"
- const int LIGODAQ\_DATATYPE\_FLOAT = 4
- const int LIGODAQ\_DATATYPE\_INT32 = 2
- const int LIGODAQ\_DATATYPE\_DEFAULT = LIGODAQ\_DATATYPE\_FLOAT
- const char \*const LIGODAQ\_UNIT\_NAME = "units"
- const char \*const LIGODAQ\_UNIT\_NONE = "none"
- const char \*const LIGODAQ\_UNIT\_DEFAULT = LIGODAQ\_UNIT\_NONE
- const char \*const LIGODAQ\_INI\_HEADER

### 7.2.1 Detailed Description

Namespace for tpy-db conversion.

[EpicsTpy](#) name space

## 7.3 InfoPlc Namespace Reference

Namespace for Info communication.

### Classes

- class [InfoInterface](#)  
*Info interface.*
- class [process\\_arg\\_info](#)  
*Arguments for processing.*

### Typedefs

- typedef bool(InfoInterface::\* [info\\_update\\_method](#)) ()  
*Pointer to info update method.*
- typedef std::tuple<ParseUtil::variable\_name, ParseUtil::process\_type\_enum, ParseUtil::opc\_list, std::stringcase, bool, [update\\_enum](#), [info\\_update\\_method](#) > [info\\_dbrecord\\_type](#)
- typedef std::vector< [info\\_dbrecord\\_type](#) > [info\\_dbrecord\\_list](#)  
*List type of db info tuples.*

### Enumerations

- enum [update\\_enum](#) { [update\\_enum::forever](#), [update\\_enum::once](#), [update\\_enum::done](#) }  
*Update frequency type.*

### 7.3.1 Detailed Description

Namespace for Info communication.

[InfoPlc](#) name space, which has all the classes and functions used for communicating with Info.

## 7.4 ParseTpy Namespace Reference

Namespace for parsing.

### Classes

- class [ads\\_routing\\_info](#)  
*ADS routing information.*
- class [base\\_record](#)  
*Base record definition.*
- class [compiler\\_info](#)  
*Compiler information.*
- class [item\\_record](#)  
*item record*
- class [parserinfo\\_type](#)  
*Parser information.*
- class [project\\_record](#)  
*Project information.*
- class [symbol\\_record](#)  
*Symbol record.*
- class [tpy\\_file](#)  
*Tpy file parsing.*
- class [type\\_map](#)  
*Type dictionary.*
- class [type\\_record](#)  
*Type record information.*

### Typedefs

- typedef std::pair< int, int > [dimension](#)
- typedef std::list< [dimension](#) > [dimensions](#)
- typedef std::map< int, std::stringcase > [enum\\_map](#)
- typedef std::pair< int, std::stringcase > [enum\\_pair](#)
- typedef std::list< [item\\_record](#) > [item\\_list](#)
- typedef std::multimap< unsigned int, [type\\_record](#) > [type\\_multipmap](#)
- typedef std::list< [symbol\\_record](#) > [symbol\\_list](#)

### Enumerations

- enum [type\\_enum](#) {  
    [type\\_enum::unknown](#), [type\\_enum::simple](#), [type\\_enum::arraytype](#), [type\\_enum::enumtype](#),  
    [type\\_enum::structtype](#), [type\\_enum::functionblock](#) }  
*Type enum.*

## Functions

- bool `compareNamesWoNamespace` (const `std::stringcase` &p1, const `std::stringcase` &p2)
- bool `get_decoration` (const char \*\*atts, unsigned int &decoration)
- bool `get_pointer` (const char \*\*atts)

## Variables

- const char \*const `xmlPlcProjectInfo` = "PlcProjectInfo"  
*PLC project info.*
- const char \*const `xmlProjectInfo` = "ProjectInfo"  
*Project info.*
- const char \*const `xmlRoutingInfo` = "RoutingInfo"  
*Routing info.*
- const char \*const `xmlCompilerInfo` = "CompilerInfo"  
*Compiler info.*
- const char \*const `xmlAdsInfo` = "AdsInfo"  
*ADS info.*
- const char \*const `xmlDataTypes` = "DataTypes"  
*Data types.*
- const char \*const `xmlDataType` = "DataType"  
*Data type.*
- const char \*const `xmlSymbols` = "Symbols"  
*Symbols.*
- const char \*const `xmlSymbol` = "Symbol"  
*Symbol.*
- const char \*const `xmlProperties` = "Properties"  
*Properties.*
- const char \*const `xmlProperty` = "Property"  
*Property.*
- const char \*const `xmlCompilerVersion` = "CompilerVersion"  
*Compiler version.*
- const char \*const `xmlTwinCATVersion` = "TwinCATVersion"  
*TwinCAT version.*
- const char \*const `xmlCpuFamily` = "CpuFamily"  
*CPU family.*
- const char \*const `xmlNetId` = "NetId"  
*Net ID.*
- const char \*const `xmlPort` = "Port"  
*Port.*
- const char \*const `xmlTargetName` = "TargetName"  
*Target name.*
- const char \*const `xmlName` = "Name"  
*Name.*
- const char \*const `xmlType` = "Type"  
*Type.*
- const char \*const `xmlAttrDecoration` = "Decoration"  
*Decoration.*
- const char \*const `xmlAttrPointer` = "Pointer"  
*Pointer.*

- const char \*const `xmlIGroup` = "IGroup"  
*I Group.*
- const char \*const `xmlOffset` = "lOffset"  
*I Offset.*
- const char \*const `xmlBitSize` = "BitSize"  
*Bit size.*
- const char \*const `xmlBitOffs` = "BitOffs"  
*Bit Offset.*
- const char \*const `xmlArrayInfo` = "ArrayInfo"  
*Array info.*
- const char \*const `xmlArrayLBound` = "LBound"  
*Lower bound.*
- const char \*const `xmlArrayElements` = "Elements"  
*Elements.*
- const char \*const `xmlSubItem` = "SubItem"  
*Sub item.*
- const char \*const `xmlFbInfo` = "FbInfo"  
*Fb info.*
- const char \*const `xmlEnumInfo` = "EnumInfo"  
*Enum info.*
- const char \*const `xmlEnumText` = "Text"  
*Text.*
- const char \*const `xmlEnumEnum` = "Enum"  
*Enum.*
- const char \*const `xmlEnumComment` = "Comment"  
*Comment.*
- const char \*const `xmlValue` = "Value"  
*Value.*
- const char \*const `xmlDesc` = "Desc"  
*Description.*
- const char \*const `opcExport` = "opc"  
*OPC.*
- const char \*const `opcProp` = "opc\_prop"  
*OPC property.*
- const char \*const `opcBracket` = "["  
*OPC bracket.*

#### 7.4.1 Detailed Description

Namespace for parsing.

Namespace for tpy parsing.

`ParseTpy` name space

#### 7.4.2 Function Documentation

#### 7.4.2.1 compareNamesWoNamespace()

```
bool ParseTpy::compareNamesWoNamespace (
    const std::stringcase & p1,
    const std::stringcase & p2 )
```

compareNamesWoNamespace

Definition at line 355 of file ParseTpy.cpp.

#### 7.4.2.2 get\_decoration()

```
bool ParseTpy::get_decoration (
    const char ** attrs,
    unsigned int & decoration )
```

XML get decoration number from attribute

Definition at line 490 of file ParseTpy.cpp.

References `xmlAttrDecoration`.

#### 7.4.2.3 get\_pointer()

```
bool ParseTpy::get_pointer (
    const char ** attrs )
```

XML get pointer from attribute

Definition at line 504 of file ParseTpy.cpp.

References `xmlAttrPointer`.

## 7.5 ParseUtil Namespace Reference

Namespace for parsing utilities.

## Classes

- class [bit\\_location](#)  
*Bit location.*
- class [memory\\_location](#)  
*Memory location.*
- class [opc\\_list](#)  
*OPC list.*
- class [optarg](#)  
*Optional arguments.*
- class [process\\_arg](#)  
*Arguments for processing.*
- class [process\\_arg\\_tc](#)  
*Arguments for processing.*
- class [replacement\\_rules](#)  
*Replacement rules.*
- class [tag\\_processing](#)  
*Tag processing selection.*
- class [variable\\_name](#)  
*Variable name.*

## TypeDefs

- typedef std::map< [std::stringcase](#), [std::stringcase](#) > [replacement\\_table](#)
- typedef std::map< int, [std::stringcase](#) > [property\\_map](#)
- typedef std::pair< int, [std::stringcase](#) > [property\\_el](#)

## Enumerations

- enum [opc\\_enum](#) { [opc\\_enum::no\\_change](#), [opc\\_enum::publish](#), [opc\\_enum::silent](#) }  
*OPC state enum.*
- enum [process\\_type\\_enum](#) {  
  [process\\_type\\_enum::pt\\_invalid](#), [process\\_type\\_enum::pt\\_int](#), [process\\_type\\_enum::pt\\_real](#), [process\\_type\\_enum::pt\\_bool](#),  
  [process\\_type\\_enum::pt\\_string](#), [process\\_type\\_enum::pt\\_enum](#), [process\\_type\\_enum::pt\\_binary](#) }  
*Process type.*
- enum [process\\_tag\\_enum](#) { [process\\_tag\\_enum::all](#), [process\\_tag\\_enum::atomic](#), [process\\_tag\\_enum::structured](#) }  
*Tag preoicessing enum.*

## Variables

- const int [OPC\\_PROP\\_CDT](#) = 1
- const int [OPC\\_PROP\\_VALUE](#) = 2
- const int [OPC\\_PROP\\_QUALITY](#) = 3
- const int [OPC\\_PROP\\_TIME](#) = 4
- const int [OPC\\_PROP\\_RIGHTS](#) = 5
- const int [OPC\\_PROP\\_SCANRATE](#) = 6
- const int [OPC\\_PROP\\_UNIT](#) = 100
- const int [OPC\\_PROP\\_DESC](#) = 101
- const int [OPC\\_PROP\\_HIEU](#) = 102

- const int `OPC_PROP_LOEU` = 103
- const int `OPC_PROP_HIRANGE` = 104
- const int `OPC_PROP_LORANGE` = 105
- const int `OPC_PROP_CLOSE` = 106
- const int `OPC_PROP_OPEN` = 107
- const int `OPC_PROP_TIMEZONE` = 108
- const int `OPC_PROP_FGC` = 201
- const int `OPC_PROP_BGC` = 202
- const int `OPC_PROP_BLINK` = 203
- const int `OPC_PROP_BMP` = 204
- const int `OPC_PROP SND` = 205
- const int `OPC_PROP_HTML` = 206
- const int `OPC_PROP_AVI` = 207
- const int `OPC_PROP_ALMSTAT` = 300
- const int `OPC_PROP_ALMHELP` = 301
- const int `OPC_PROP_ALMAREAS` = 302
- const int `OPC_PROP_ALMPRIMARYAREA` = 303
- const int `OPC_PROP_ALMCONDITION` = 304
- const int `OPC_PROP_ALMLIMIT` = 305
- const int `OPC_PROP_ALMDB` = 306
- const int `OPC_PROP_ALMHH` = 307
- const int `OPC_PROP_ALMH` = 308
- const int `OPC_PROP_ALML` = 309
- const int `OPC_PROP_ALMLL` = 310
- const int `OPC_PROP_ALMROC` = 311
- const int `OPC_PROP_ALMDEV` = 312
- const int `OPC_PROP_PREC` = 8500
- const int `OPC_PROP_ZRST` = 8510
- const int `OPC_PROP_FFST` = 8525
- const int `OPC_PROP_RECTYPE` = 8600
- const int `OPC_PROP_INOUT` = 8601
- const char \*const `OPC_PROP_INPUT` = "input"
- const char \*const `OPC_PROP_OUTPUT` = "output"
- const int `OPC_PROP_TSE` = 8602
- const int `OPC_PROP_PINI` = 8603
- const int `OPC_PROP_DTYP` = 8604
- const int `OPC_PROP_SERVER` = 8610
- const int `OPC_PROP_PLNAME` = 8611
- const int `OPC_PROP_ALIAS` = 8620
- const int `OPC_PROP_ALMOSV` = 8700
- const int `OPC_PROP_ALMZSV` = 8701
- const int `OPC_PROP_ALMCOSV` = 8702
- const int `OPC_PROP_ALMUNSV` = 8703
- const int `OPC_PROP_ALMZRSV` = 8710
- const int `OPC_PROP_ALMFFSV` = 8725
- const int `OPC_PROP_ALMHHSV` = 8727
- const int `OPC_PROP_ALMHHSV` = 8728
- const int `OPC_PROP_ALMLSV` = 8729
- const int `OPC_PROP_ALMLLSV` = 8730
- const int `OPC_PROP_FIELD_BEG` = 8800
- const int `OPC_PROP_FIELD_END` = 9000

### 7.5.1 Detailed Description

Namespace for parsing utilities.

[ParseUtil](#) name space

[ParseTyp](#) name space

## 7.6 plc Namespace Reference

Namespace for abstract plc functionality.

### Classes

- class [BasePLC](#)  
*Base PLC.*
- class [BaseRecord](#)  
*Class for managing a tag/channel.*
- class [DataValue](#)  
*Data value.*
- struct [DataValueTraits](#)  
*Data value traits.*
- struct [DataValueTypeDef](#)  
*Collection of type definitions.*
- class [Interface](#)  
*Abstract interface.*
- struct [scanner\\_thread\\_args](#)  
*Scanner thread arguments.*
- class [System](#)  
*System to keep track of PLCs.*

### Typedefs

- typedef std::shared\_ptr< [BasePLC](#) > [BasePLCPtr](#)  
*Smart pointer to PLC.*
- typedef std::unique\_ptr< [Interface](#) > [InterfacePtr](#)  
*Smart pointer to interface.*
- typedef std::shared\_ptr< [BaseRecord](#) > [BaseRecordPtr](#)  
*smart pointer to record*
- typedef std::unordered\_map< std::stringcase, [BaseRecordPtr](#) > [BaseRecordList](#)  
*list of record*
- typedef std::map< std::stringcase, [BasePLCPtr](#) > [BasePLCList](#)  
*BasePLC map.*

## Enumerations

- enum `data_type_enum` {
 `data_type_enum::dtInvalid, data_type_enum::dtBool, data_type_enum::dtInt8, data_type_enum::dtUInt8,`
`data_type_enum::dtInt16, data_type_enum::dtUInt16, data_type_enum::dtInt32, data_type_enum::dtUInt32,`
`data_type_enum::dtInt64, data_type_enum::dtUInt64, data_type_enum::dtFloat, data_type_enum::dtDouble,`
`data_type_enum::dtString, data_type_enum::dtWString, data_type_enum::dtBinary }`

*Data type enumeration.*
  - enum `access_rights_enum` { `access_rights_enum::read_only, access_rights_enum::write_only, access_rights_enum::read_write` }
- Access rights enum.*

## Functions

- template<> bool `reset_and_read` (`DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::type_wstring &dest, DataValueTypeDef::atomic_string *source)`

*Reset and read.*
- template<> bool `write_and_test` (`DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, DataValueTypeDef::atomic_wstring *dest, const DataValueTypeDef::type_string &source)`

*Write and test.*
- VOID CALLBACK `ScannerProc` (LPVOID `lpArg`, DWORD `dwTimerLowValue`, DWORD `dwTimerHighValue`)
 

*Scanner thread callback.*
- DWORD WINAPI `scannerThread` (`scanner_thread_args args`)
 

*Scanner thread.*
- template<typename T, typename U>
 bool `reset_and_read` (`DataValueTypeDef::atomic_bool &dirty, T &dest, U source)`

*Reset and read.*
- template<typename T>
 bool `reset_and_read` (`DataValueTypeDef::atomic_bool &dirty, T &dest, typename DataValueTraits< T >::traits_atomic *source)`

*Reset and read.*
- template<typename T, typename U>
 bool `write_and_test` (`DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, U dest, const T &source)`

*Write and test.*
- template<typename T>
 bool `write_and_test` (`DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, typename DataValueTraits< T >::traits_atomic *dest, const T &source)`

*Write and test.*

### 7.6.1 Detailed Description

Namespace for abstract plc functionality.

PLC namespace

### 7.6.2 Typedef Documentation

### 7.6.2.1 BasePLCList

```
typedef std::map<std::stringcase, BasePLCPtr> plc::BasePLCList
```

BasePLC map.

This is list of [BasePLC](#), ordered by their name.

Definition at line 893 of file plcBase.h.

### 7.6.2.2 BasePLCPtr

```
typedef std::shared_ptr<BasePLC> plc::BasePLCPtr
```

Smart pointer to PLC.

This is a smart pointer to a PLC

Definition at line 26 of file plcBase.h.

### 7.6.2.3 BaseRecordList

```
typedef std::unordered_map<std::stringcase, BaseRecordPtr> plc::BaseRecordList
```

list of record

This is a list of tag/channel records organized as a hash map

Definition at line 710 of file plcBase.h.

### 7.6.2.4 BaseRecordPtr

```
typedef std::shared_ptr<BaseRecord> plc::BaseRecordPtr
```

smart pointer to record

This is a smart pointer to a tag/channel record

Definition at line 705 of file plcBase.h.

### 7.6.2.5 InterfacePtr

```
typedef std::unique_ptr<Interface> plc::InterfacePtr
```

Smart pointer to interface.

This is a smart pointer for [Interface](#)

Definition at line 69 of file plcBase.h.

## 7.6.3 Enumeration Type Documentation

### 7.6.3.1 access\_rights\_enum

```
enum plc::access_rights_enum [strong]
```

Access rights enum.

Enum for access rights of a record

## Enumerator

|            |             |
|------------|-------------|
| read_only  | Read only.  |
| write_only | Write only. |
| read_write | Read/write. |

Definition at line 479 of file plcBase.h.

### 7.6.3.2 data\_type\_enum

```
enum plc::data\_type\_enum [strong]
```

Data type enumeration.

This is an enumerated type listing all the available data types

## Enumerator

|           |  |
|-----------|--|
| dtInvalid | Invalid data type.                     |
| dtBool    | Boolean.                               |
| dtInt8    | 1-byte integer                         |
| dtUInt8   | 1-byte unsigned integer                |
| dtInt16   | 2-byte integer                         |
| dtUInt16  | 2-byte unsigned integer                |
| dtInt32   | 4-byte integer                         |
| dtUInt32  | 4-byte unsigned integer                |
| dtInt64   | 8-byte integer                         |
| dtUInt64  | 8-byte unsigned integer                |
| dtFloat   | 4-byte single precision floating point |
| dtDouble  | 8-byte double precision floating point |
| dtString  | string class                           |
| dtWString | wstring class                          |
| dtBinary  | binary object                          |

Definition at line 74 of file plcBase.h.

## 7.6.4 Function Documentation

### 7.6.4.1 reset\_and\_read() [1/3]

```
template<>
bool plc::reset\_and\_read (
```

```

    DataValueTypeDef::atomic_bool & dirty,
    DataValueTypeDef::type_wstring & dest,
    DataValueTypeDef::atomic_string * source )

```

Reset and read.

Will read the value and reset the dirty flag (wstring from string).

Definition at line 71 of file plcBase.cpp.

References plc::DataValueTypeDef::memory\_order.

Referenced by plc::DataValue::Read().

Here is the caller graph for this function:



#### 7.6.4.2 reset\_and\_read() [2/3]

```

template<typename T >
bool plc::reset_and_read (
    DataValueTypeDef::atomic_bool & dirty,
    T & dest,
    typename DataValueTraits< T >::traits_atomic * source )

```

Reset and read.

Will read the value and reset the dirt flag (same type).

Definition at line 34 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory\_order.

#### 7.6.4.3 reset\_and\_read() [3/3]

```

template<typename T , typename U >
bool plc::reset_and_read (
    DataValueTypeDef::atomic_bool & dirty,
    T & dest,
    U source )

```

Reset and read.

Will read the value and reset the dirty flag.

Definition at line 21 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory\_order.

#### 7.6.4.4 ScannerProc()

```
VOID CALLBACK plc::ScannerProc (
    LPVOID lpArg,
    DWORD dwTimerLowValue,
    DWORD dwTimerHighValue )
```

Scanner thread callback.

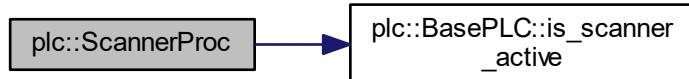
Scanner thread callback with periodic timer

Definition at line 675 of file plcBase.cpp.

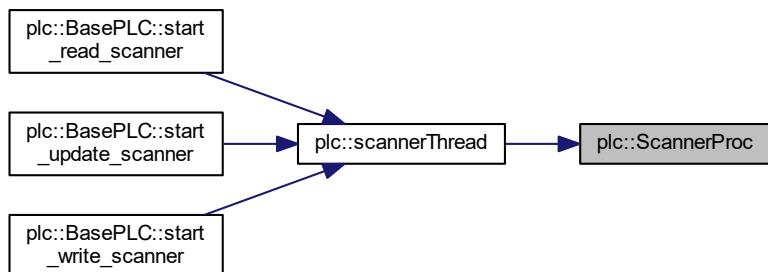
References `plc::BasePLC::is_scanner_active()`, `plc::scanner_thread_args::plc`, and `plc::scanner_thread_args::scanner`.

Referenced by `scannerThread()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 7.6.4.5 scannerThread()

```
DWORD WINAPI plc::scannerThread (
    scanner_thread_args args )
```

Scanner thread.

Scanner thread with periodic timer This function uses the windows waitable timer which will call a completion routine at a regular interval. The completion routine in this case is one of either read\_scanner, write\_scanner, or update\_scanner.

Definition at line 694 of file plcBase.cpp.

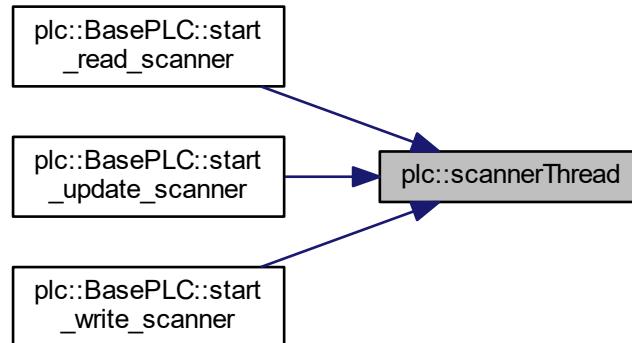
References ScannerProc(), and plc::scanner\_thread\_args::scanperiod.

Referenced by plc::BasePLC::start\_read\_scanner(), plc::BasePLC::start\_update\_scanner(), and plc::BasePLC::start\_write\_scanner().

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.6.4.6 write\_and\_test() [1/3]

```
template<>
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    DataValueTypeDef::atomic_wstring * dest,
    const DataValueTypeDef::type_string & source )
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one (string to wstring).

Definition at line 88 of file plcBase.cpp.

References plc::DataValueTypeDef::memory\_order.

Referenced by plc::DataValue::Write().

Here is the caller graph for this function:



### 7.6.4.7 write\_and\_test() [2/3]

```
template<typename T >
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    typename DataValueTraits< T >::traits_atomic * dest,
    const T & source )
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one (same type).

Definition at line 68 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory\_order.

#### 7.6.4.8 `write_and_test()` [3/3]

```
template<typename T , typename U >
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    U dest,
    const T & source )
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one.

Definition at line 48 of file plcBaseTemplate.h.

References `plc::DataValueTypeDef::memory_order`.

## 7.7 TcComms Namespace Reference

Namespace for TCat communication.

### Classes

- class [AmsRouterNotification](#)  
*AMS Router Notification.*
- struct [DataPar](#)  
*Memory location struct.*
- class [TCatInterface](#)  
*TCat interface class.*
- class [TcPLC](#)  
*TwinCAT PLC.*
- class [tcProcWrite](#)  
*TwinCAT process write requests.*

### Functions

- void [errorPrintf](#) (int nErr)  
*errorPrintf*
- void \_\_stdcall [ADSCallback](#) (AmsAddr \*pAddr, AdsNotificationHeader \*pNotification, unsigned long plcid)
- void \_\_stdcall [RouterCall](#) (long nReason)

## Variables

- const int **MAX\_REQ\_SIZE** = 250000  
*maximum allowed request size (bytes)*
- const int **MAX\_SINGLE\_GAP\_SIZE** = 50  
*maximum allowed size (bytes) of a memory gap within continuous request*
- const double **MAX\_REL\_GAP** = 0.25  
*(maximum allowed total gap size) / (current request size)*
- const int **MIN\_REL\_GAP\_SIZE** = 100  
*minimum allowed relative gap size (bytes)*
- const int **default\_scanrate** = 100  
*default PLC TwinCAT scan rate (100ms)*
- const int **minimum\_scanrate** = 5  
*minimum PLC TwinCAT scan rate (5ms)*
- const int **maximum\_scanrate** = 10000  
*maximum PLC TwinCAT scan rate (10s)*
- const int **default\_multiple** = 10  
*default multiple for PLC EPICS scan rate (10)*
- const int **minimum\_multiple** = 1  
*minimum multiple for PLC EPICS scan rate (1)*
- const int **maximum\_multiple** = 200  
*maximum multiple for PLC EPICS scan rate (200)*

### 7.7.1 Detailed Description

Namespace for TCat communication.

[TcComms](#) name space, which has all the classes and functions used for communicating with TCat.

### 7.7.2 Function Documentation

#### 7.7.2.1 **ADScallback()**

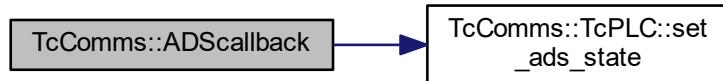
```
void __stdcall TcComms::ADScallback (
    AmsAddr * pAddr,
    AdsNotificationHeader * pNotification,
    unsigned long plcId )
```

Callback for ADS state change

Definition at line 654 of file tcComms.cpp.

References TcComms::TcPLC::set\_ads\_state().

Here is the call graph for this function:



### 7.7.2.2 errorPrintf()

```
void TcComms::errorPrintf (
    int nErr )
```

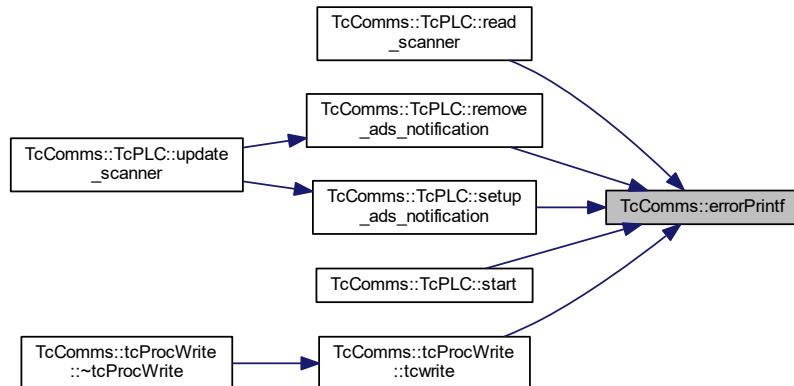
**errorPrintf**

Print an error message for an ADS error return code

Definition at line 30 of file tcComms.cpp.

Referenced by TcComms::TcPLC::read\_scanner(), TcComms::TcPLC::remove\_ads\_notification(), TcComms::TcPLC::setup\_ads\_notification(), TcComms::TcPLC::start(), and TcComms::tcProcWrite::tcwrite().

Here is the caller graph for this function:



### 7.7.2.3 RouterCall()

```
void __stdcall TcComms::RouterCall (
    long nReason )
```

**Callback for AMS router state change**

Definition at line 883 of file tcComms.cpp.

# Chapter 8

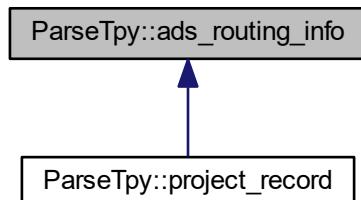
## Class Documentation

### 8.1 ParseTpy::ads\_routing\_info Class Reference

ADS routing information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::ads\_routing\_info:



#### Public Member Functions

- `ads_routing_info ()`  
*Default constructor.*
- `ads_routing_info (const std::string& netid, int port=801)`  
*Constructor.*
- `ads_routing_info (const std::string& netid, int port, const std::string& targetname)`  
*Constructor.*
- `const std::string& get_netid () const`  
*Get ADS net id.*
- `void set_netid (const std::string& netid)`  
*Set ADS net id.*
- `int get_port () const`

- `void set_port (int port)`  
*Set ADS port.*
- `const std::stringcase & get_targetname () const`  
*Get ADS target name.*
- `void set_targetname (const std::stringcase &targetname)`  
*Set ADS target name.*
- `bool isValid () const`  
*Checks, if net id is of the form n.n.n.n.n.n.*
- `std::stringcase get () const`
- `bool set (const std::stringcase &s)`
- `bool get (unsigned char &a1, unsigned char &a2, unsigned char &a3, unsigned char &a4, unsigned char &a5, unsigned char &a6) const`

## Protected Attributes

- `std::stringcase ads_netid`  
*ADS net ID.*
- `int ads_port`  
*ADS port.*
- `std::stringcase ads_targetname`  
*ADS target name.*

### 8.1.1 Detailed Description

ADS routing information.

This is a base class for storing the ADS routing information

Definition at line 23 of file ParseTpy.h.

### 8.1.2 Member Function Documentation

#### 8.1.2.1 `get()` [1/2]

```
std::stringcase ParseTpy::ads_routing_info::get ( ) const
```

Gets a string representation of a ads routing information

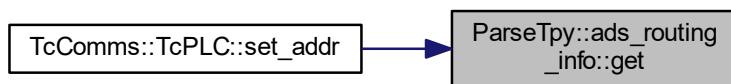
**Returns**

string with format "tc://netid:port/", empty on error

Definition at line 215 of file ParseTpy.cpp.

Referenced by TcComms::TcPLC::set\_addr().

Here is the caller graph for this function:



### 8.1.2.2 get() [2/2]

```
bool ParseTpy::ads_routing_info::get (
    unsigned char & a1,
    unsigned char & a2,
    unsigned char & a3,
    unsigned char & a4,
    unsigned char & a5,
    unsigned char & a6 ) const
```

Get address in net format

**Parameters**

|           |                                   |
|-----------|-----------------------------------|
| <i>a1</i> | First address qualifier (return)  |
| <i>a2</i> | Second address qualifier (return) |
| <i>a3</i> | Third address qualifier (return)  |
| <i>a4</i> | Fourth address qualifier (return) |
| <i>a5</i> | Fifth address qualifier (return)  |
| <i>a6</i> | Sixth address qualifier (return)  |

Definition at line 228 of file ParseTpy.cpp.

### 8.1.2.3 set()

```
bool ParseTpy::ads_routing_info::set (
    const std::string& s )
```

Set the ads routing information using a string of the form: "tc://netid:port/" where netid is a string of the format n.n.n.n.n

#### Parameters

|                |   |
|----------------|---|
| <code>s</code> | String describing the ads routing information |
|----------------|---|

#### Returns

True if successful

Definition at line 249 of file ParseTpy.cpp.

The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

## 8.2 TcComms::AmsRouterNotification Class Reference

AMS Router Notification.

```
#include <tcComms.h>
```

### Public Member Functions

- int `get_ads_version () const`  
*get ADS protocol/library version*
- int `get_ads_revision () const`  
*get ADS protocol/library revision*
- int `get_ads_build () const`  
*get ADS protocol/library build*

### Static Public Member Functions

- static AmsRouterEvent `get_router_notification ()`  
*get router notification*
- static const `AmsRouterNotification & get_instance ()`  
*get global instance*

### Protected Attributes

- int `ads_version`  
*ADS protocol/library version.*
- int `ads_revision`  
*ADS protocol/library revision.*
- int `ads_build`  
*ADS protocol/library build.*

## Friends

- void \_\_stdcall RouterCall (long)  
*Notification callback is a friend.*

### 8.2.1 Detailed Description

AMS Router Notification.

Class for a AMS router notifications

Definition at line 411 of file tcComms.h.

### 8.2.2 Friends And Related Function Documentation

#### 8.2.2.1 RouterCall

```
void __stdcall RouterCall (
    long nReason ) [friend]
```

Notification callback is a friend.

Callback for AMS router state change

Definition at line 883 of file tcComms.cpp.

The documentation for this class was generated from the following files:

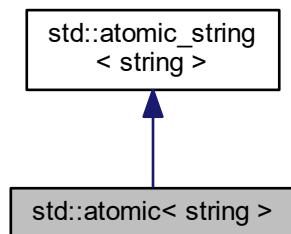
- [tcComms.h](#)
- [tcComms.cpp](#)

## 8.3 std::atomic< string > Class Reference

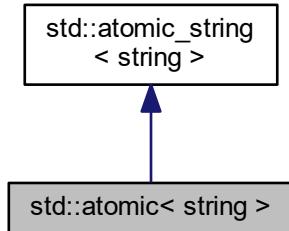
[atomic<string>](#)

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< string >:



Collaboration diagram for std::atomic< string >:



## Public Member Functions

- [atomic \(\)](#)  
*Default constructor.*
- [atomic \(const string &s\)](#)  
*Constructor from string.*

## Additional Inherited Members

### 8.3.1 Detailed Description

#### [atomic<string>](#)

This is a class implements an atomic specialization for std::string

Definition at line 71 of file atomic\_string.h.

The documentation for this class was generated from the following file:

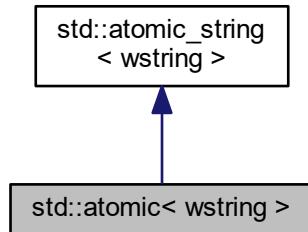
- [atomic\\_string.h](#)

## 8.4 std::atomic< wstring > Class Reference

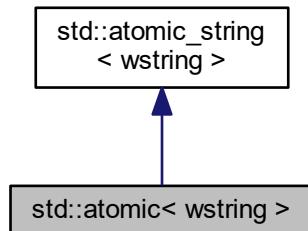
#### [atomic<wstring>](#)

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< wstring >:



Collaboration diagram for std::atomic< wstring >:



## Public Member Functions

- [atomic \(\)](#)  
*Default constructor.*
- [atomic \(const wstring &s\)](#)  
*Constructor from string.*

## Additional Inherited Members

### 8.4.1 Detailed Description

#### [atomic<wstring>](#)

This is a class implements an atomic specialization for std::wstring

Definition at line 84 of file atomic\_string.h.

The documentation for this class was generated from the following file:

- [atomic\\_string.h](#)

## 8.5 std::atomic\_string< stringT > Class Template Reference

atomic strings

```
#include <atomic_string.h>
```

### Public Member Functions

- **atomic\_string ()**  
*Default constructor.*
- **atomic\_string (const stringT &s)**  
*Constructor from data.*
- **stringT operator= (const stringT &right)**  
*Assignment operator on basic type.*
- **bool is\_lock\_free () const**  
*Not lock free.*
- **void store (stringT value, memory\_order order=memory\_order\_seq\_cst)**  
*Store.*
- **stringT load (memory\_order order=memory\_order\_seq\_cst) const**  
*Load.*
- **operator stringT () const**  
*Convert to string.*
- **stringT exchange (const stringT &value, memory\_order order=memory\_order\_seq\_cst)**  
*Exchange.*
- **bool compare\_exchange\_weak (stringT &exp, const stringT &value, memory\_order order1, memory\_order order2)**  
*Compare exchange.*
- **bool compare\_exchange\_weak (stringT &exp, const stringT &value, memory\_order order=memory\_order\_seq\_cst)**  
*Compare exchange.*
- **bool compare\_exchange\_strong (stringT &exp, const stringT &value, memory\_order order1, memory\_order order2)**  
*Compare exchange.*
- **bool compare\_exchange\_strong (stringT &exp, const stringT &value, memory\_order order=memory\_order\_seq\_cst)**  
*Compare exchange.*

### Protected Attributes

- **atomic\_flag flag**  
*flag for spin lock*
- **stringT data**  
*data string*

#### 8.5.1 Detailed Description

```
template<typename stringT>
class std::atomic_string< stringT >
```

atomic strings

This is a class implements an atomic specialization for strings

Definition at line 16 of file atomic\_string.h.

## 8.5.2 Member Function Documentation

### 8.5.2.1 operator=()

```
template<typename stringT >
stringT std::atomic_string< stringT >::operator= (
    const stringT & right )
```

Assignment operator on basic type.

Assignment operator.

Definition at line 96 of file atomic\_string.h.

The documentation for this class was generated from the following file:

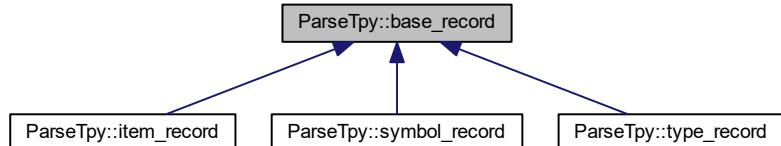
- [atomic\\_string.h](#)

## 8.6 ParseTpy::base\_record Class Reference

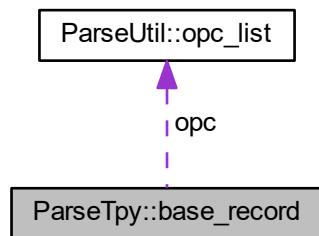
Base record definition.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::base\_record:



Collaboration diagram for ParseTpy::base\_record:



## Public Member Functions

- `base_record ()`  
`Default constructor.`
- `base_record (const std::stringcase &n)`
- `base_record (const std::stringcase &n, const ParseUtil::opc_list &o)`
- `base_record (const std::stringcase &n, const ParseUtil::opc_list &o, const std::stringcase &tn, int td=0)`
- `base_record (const std::stringcase &n, const std::stringcase &tn, int td=0)`
- `const std::stringcase & get_name () const`  
`Get name.`
- `std::stringcase & get_name ()`  
`Get name.`
- `void set_name (std::stringcase n)`  
`Set name.`
- `const std::stringcase & get_type_name () const`  
`Get type name.`
- `std::stringcase & get_type_name ()`  
`Get type name.`
- `void set_type_name (std::stringcase t)`  
`Set type name.`
- `unsigned int get_type_decoration () const`  
`Get type decoration.`
- `void set_type_decoration (unsigned int id)`  
`Set type decoration.`
- `bool get_type_pointer () const`  
`Get type pointer.`
- `void set_type_pointer (bool isPointer)`  
`Set type pointer.`
- `const ParseUtil::opc_list & get_opc () const`  
`Get OPC list.`
- `ParseUtil::opc_list & get_opc ()`  
`Get OPC list.`

## Protected Attributes

- `std::stringcase name`  
`name of type`
- `std::stringcase type_n`  
`type definition`
- `unsigned int type_decoration`  
`decoration or type ID of type definition`
- `bool type_pointer`  
`this is a pointer`
- `ParseUtil::opc_list opc`  
`list of opc properties`

### 8.6.1 Detailed Description

Base record definition.

This is a base class for storing name, type, type id and opc list

Definition at line 149 of file ParseTpy.h.

## 8.6.2 Constructor & Destructor Documentation

### 8.6.2.1 base\_record() [1/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n ) [inline], [explicit]
```

Constructor

Parameters

|          |      |
|----------|------|
| <i>n</i> | Name |
|----------|------|

Definition at line 156 of file ParseTpy.h.

### 8.6.2.2 base\_record() [2/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
    const ParseUtil::opc_list & o ) [inline]
```

Constructor

Parameters

|          |          |
|----------|----------|
| <i>n</i> | Name     |
| <i>o</i> | OPC list |

Definition at line 161 of file ParseTpy.h.

### 8.6.2.3 base\_record() [3/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
    const ParseUtil::opc_list & o,
    const std::stringcase & tn,
    int td = 0 ) [inline]
```

Constructor

Parameters

|           |                        |
|-----------|------------------------|
| <i>n</i>  | Name                   |
| <i>o</i>  | OPC list               |
| <i>tn</i> | Type name              |
| <i>td</i> | Type decortation or id |

---

Definition at line 168 of file ParseTpy.h.

#### 8.6.2.4 base\_record() [4/4]

```
ParseTpy::base_record::base_record (
    const std::string& n,
    const std::string& tn,
    int td = 0 ) [inline]
```

Constructor

##### Parameters

|           |                        |
|-----------|------------------------|
| <i>n</i>  | Name                   |
| <i>tn</i> | Type name              |
| <i>td</i> | Type decortation or id |

Definition at line 175 of file ParseTpy.h.

The documentation for this class was generated from the following file:

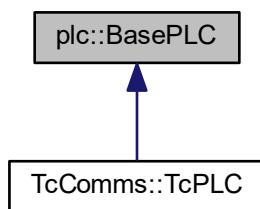
- [ParseTpy.h](#)

## 8.7 plc::BasePLC Class Reference

Base PLC.

```
#include <plcBase.h>
```

Inheritance diagram for plc::BasePLC:



## Public Types

- `typedef std::recursive_mutex mutex_type`  
*Defines the mutex type.*
- `typedef std::lock_guard< mutex_type > guard`  
*Defined the mutex guard type.*
- `typedef DataValueTypeDef::type_uint64 time_type`  
*Define timestamp type.*
- `typedef void(BasePLC::* scanner_func) ()`  
*Function pointer to scanner.*

## Public Member Functions

- `BasePLC ()`  
*Default constructor.*
- `virtual ~BasePLC ()`  
*Destructor.*
- `int get_read_scanner_period () const`  
*Get read scanner period in ms.*
- `void set_read_scanner_period (int period)`  
*Set read scanner period in ms.*
- `bool start_read_scanner ()`  
*Start read scanner.*
- `bool terminate_read_scanner ()`  
*Terminate read scanner.*
- `int get_write_scanner_period () const`  
*Get write scanner period in ms.*
- `void set_write_scanner_period (int period)`  
*Set write scanner period in ms.*
- `bool start_write_scanner ()`  
*Start write scanner.*
- `bool terminate_write_scanner ()`  
*Terminate write scanner.*
- `int get_update_scanner_period () const`  
*Get update scanner period in ms.*
- `void set_update_scanner_period (int period)`  
*Set update scanner period in ms.*
- `bool start_update_scanner ()`  
*Start update scanner.*
- `bool terminate_update_scanner ()`  
*Terminate update scanner.*
- `bool is_scanner_active () const`  
*is scanner active?*
- `void set_scanners_active (bool active)`  
*set scanner active state*
- `void reserve (BaseRecordList::size_type n)`
- `bool add (BaseRecord *precord)`
- `bool add (BaseRecordPtr precord)`
- `BaseRecordPtr find (const std::stringcase &name)`
- `bool erase (const std::stringcase &name)`
- `bool get_next (BaseRecordPtr &next, const BaseRecordPtr &prev) const`

- template<typename func >  
void **for\_each** (func &f)
- int **count** () const  
*Count the number of records.*
- virtual void **printAllRecords** ()  
*Print all records and vals to stdout. (override for action)*
- virtual void **printRecord** (const std::string &var)
- virtual **time\_type get\_timestamp** () const  
*Get time stamp.*
- virtual **time\_t get\_timestamp\_unix** () const
- virtual void **set\_timestamp** (**time\_type** tstamp)  
*Set time stamp.*
- virtual void **update\_timestamp** ()  
*Set time stamp to current time.*
- const **std::stringcase & get\_name** () const  
*Get name.*
- const **std::stringcase & get\_alias** () const  
*Get nick name/alias.*
- void **set\_alias** (const **std::stringcase &nickname**)  
*Set nick name/alias.*
- virtual bool **start** ()
- virtual void **user\_data\_set\_valid** (bool valid)
- virtual void **plc\_data\_set\_valid** (bool valid)

## Protected Member Functions

- void **set\_name** (const **std::stringcase &n**)  
*Set name (careful! This is used for indexing in the PLCList of System)*
- virtual void **read\_scanner** ()  
*read scanner (override for action)*
- virtual void **write\_scanner** ()  
*write scanner (override for action)*
- virtual void **update\_scanner** ()  
*update scanner (override for action)*

## Protected Attributes

- **mutex\_type mux**  
*Mutex to synchronize access to this class.*
- **std::stringcase name**  
*Name.*
- **std::stringcase alias**  
*Nick name or alias (used to generate info record names)*
- **BaseRecordList records**
- **time\_type timestamp**  
*Time stamp.*
- int **read\_scanner\_period**  
*read scanner period in ms*
- int **write\_scanner\_period**  
*write scanner period in ms*

- int `update_scanner_period`  
*update scanner period in ms*
- std::atomic< bool > `scanners_active`  
*scanners are active*
- std::thread `read_thread`  
*read thread*
- std::thread `write_thread`  
*write thread*
- std::thread `update_thread`  
*update thread*

### 8.7.1 Detailed Description

Base PLC.

This is a base class for interfacing a programmable logic controller. It contains and manages a list of tag/channel records. This is a base class which needs to be used a derived class by a real implementation.

This class is MT safe and uses a mutex to synchronize access.

Definition at line 720 of file plcBase.h.

### 8.7.2 Member Function Documentation

#### 8.7.2.1 add() [1/2]

```
bool plc::BasePLC::add (
    BaseRecord * precord ) [inline]
```

Add a new tag/channel record. Adding a duplicate is not possible.

##### Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>precord</code> | Pointer to record. Will be adopted |
|----------------------|------------------------------------|

##### Returns

true, if it could be added

Definition at line 784 of file plcBase.h.

### 8.7.2.2 add() [2/2]

```
bool plc::BasePLC::add (
    BaseRecordPtr precord )
```

Add a new tag/channel record. Adding a duplicate is not possible.

**Parameters**

|                |                          |
|----------------|--------------------------|
| <i>precord</i> | Smart pointer to record. |
|----------------|--------------------------|

**Returns**

true, if it could be added

Definition at line 549 of file plcBase.cpp.

References mux, and records.

### 8.7.2.3 `erase()`

```
bool plc::BasePLC::erase (
    const std::string& name )
```

Erase a tag/channel record.

**Parameters**

|             |                |
|-------------|----------------|
| <i>name</i> | Name of record |
|-------------|----------------|

**Returns**

true if erased

Definition at line 577 of file plcBase.cpp.

References name, and records.

### 8.7.2.4 `find()`

```
BaseRecordPtr plc::BasePLC::find (
    const std::string& name )
```

Find a new tag/channel record.

**Parameters**

|             |                |
|-------------|----------------|
| <i>name</i> | Name of record |
|-------------|----------------|

**Returns**

Smart pointer to record (contains nullptr when not found)

Definition at line 563 of file plcBase.cpp.

References mux, name, and records.

**8.7.2.5 for\_each()**

```
template<typename func >
void plc::BasePLC::for_each (
    func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

**Parameters**

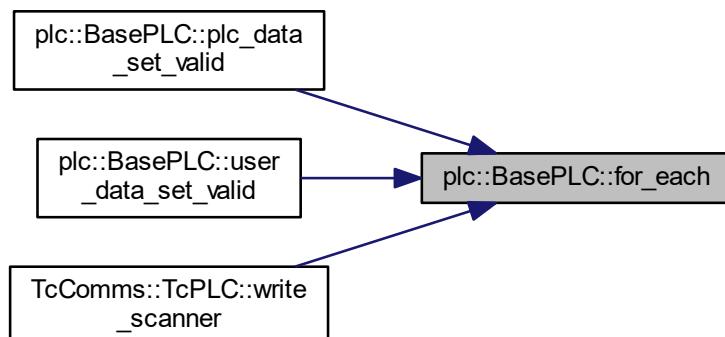
|          |  |
|----------|--|
| <i>f</i> | Function which takes BaseRecord* as the argument |
|----------|--|

Definition at line 205 of file plcBaseTemplate.h.

References mux, and records.

Referenced by plc\_data\_set\_valid(), user\_data\_set\_valid(), and TcComms::TcPLC::write\_scanner().

Here is the caller graph for this function:



### 8.7.2.6 get\_next()

```
bool plc::BasePLC::get_next (
    BaseRecordPtr & next,
    const BaseRecordPtr & prev ) const
```

Get next record in list. Resets to the beginning, if the list changes between subsequent access. This is an MT safe access method to cycle through the list. However, it is not high performance and is meant for slow external access.

#### Parameters

|             |                                  |
|-------------|----------------------------------|
| <i>next</i> | Next tag/channel record (return) |
| <i>prev</i> | tag/channel record               |

#### Returns

true if successful

Definition at line 585 of file plcBase.cpp.

References mux, and records.

Referenced by TcComms::TcPLC::update\_scanner().

Here is the caller graph for this function:



### 8.7.2.7 get\_timestamp\_unix()

```
time_t plc::BasePLC::get_timestamp_unix ( ) const [virtual]
```

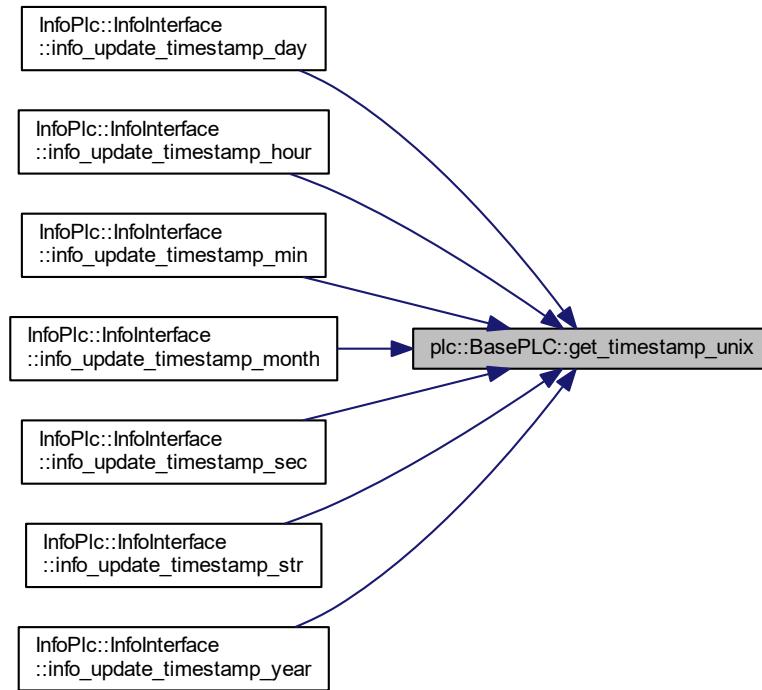
Get time stamp as unix time (seconds since 1970-01-01 00:00:00) Does not include leap seconds

Definition at line 609 of file plcBase.cpp.

References timestamp.

Referenced by InfoPlc::InfoInterface::info\_update\_timestamp\_day(), InfoPlc::InfoInterface::info\_update\_timestamp\_hour(), InfoPlc::InfoInterface::info\_update\_timestamp\_min(), InfoPlc::InfoInterface::info\_update\_timestamp\_month(), InfoPlc::InfoInterface::info\_update\_timestamp\_sec(), InfoPlc::InfoInterface::info\_update\_timestamp\_str(), and InfoPlc::InfoInterface::info\_update\_timestamp\_year().

Here is the caller graph for this function:



### 8.7.2.8 plc\_data\_set\_valid()

```
void plc::BasePLC::plc_data_set_valid (
    bool valid ) [virtual]
```

Set the valid flag for all data values by the plc

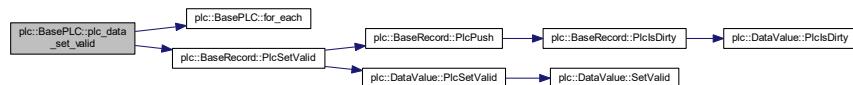
#### Parameters

|                    |   |
|--------------------|---|
| <code>valid</code> | Valid flag, true for valid, false for invalid |
|--------------------|---|

Definition at line 651 of file plcBase.cpp.

References for\_each(), and plc::BaseRecord::PlcSetValid().

Here is the call graph for this function:



### 8.7.2.9 printRecord()

```
virtual void plc::BasePLC::printRecord (
    const std::string & var ) [inline], [virtual]
```

Print a record values to stdout. (override for action)

#### Parameters

|            |                                   |
|------------|-----------------------------------|
| <i>var</i> | variable name (accepts wildcards) |
|------------|-----------------------------------|

Reimplemented in [TcComms::TcPLC](#).

Definition at line 819 of file plcBase.h.

### 8.7.2.10 reserve()

```
void plc::BasePLC::reserve (
    BaseRecordList::size_type n ) [inline]
```

Reserves the given number of elements in the tag/channel list. Use this function when you know many elements are added beforehand to avoid unnecessary rehashing.

#### Parameters

|          |  |
|----------|--|
| <i>n</i> | Number of expected tag/channel records |
|----------|--|

Definition at line 779 of file plcBase.h.

References records.

### 8.7.2.11 start()

```
virtual bool plc::BasePLC::start ( ) [inline], [virtual]
```

This function is called by the driver support in drvtc.cpp, after the PLC has been initialized. This function is overridden by the derived PLCs, and generally will start all the scanner threads.

#### Returns

true if successful

Reimplemented in [TcComms::TcPLC](#).

Definition at line 843 of file plcBase.h.

### 8.7.2.12 user\_data\_set\_valid()

```
void plc::BasePLC::user_data_set_valid (
    bool valid ) [virtual]
```

Set the valid flag for all data values by the user

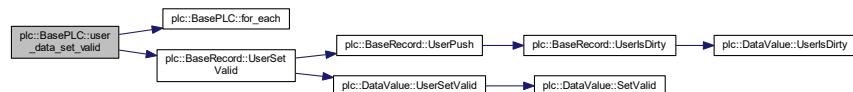
#### Parameters

|                    |   |
|--------------------|---|
| <code>valid</code> | Valid flag, true for valid, false for invalid |
|--------------------|---|

Definition at line 641 of file plcBase.cpp.

References for\_each(), and plc::BaseRecord::UserSetValid().

Here is the call graph for this function:



## 8.7.3 Member Data Documentation

### 8.7.3.1 records

`BaseRecordList` plc::BasePLC::records [protected]

List of tags/channels. The load factor is initialized to 0.5.

Definition at line 864 of file plcBase.h.

Referenced by add(), BasePLC(), count(), erase(), find(), for\_each(), get\_next(), TcComms::TcPLC::printAll← Records(), TcComms::TcPLC::printRecord(), TcComms::TcPLC::read\_scanner(), reserve(), TcComms::TcPLC← :start(), and TcComms::TcPLC::update\_scanner().

The documentation for this class was generated from the following files:

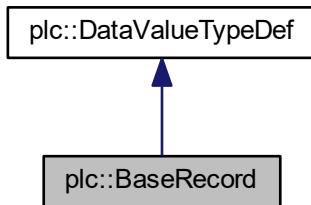
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

## 8.8 plc::BaseRecord Class Reference

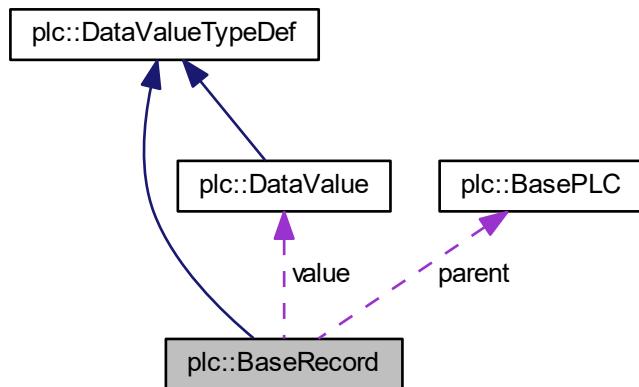
Class for managing a tag/channel.

```
#include <plcBase.h>
```

Inheritance diagram for plc::BaseRecord:



Collaboration diagram for plc::BaseRecord:



### Public Member Functions

- [BaseRecord \(\)](#)  
*Default constructor.*
- [BaseRecord \(const std::string& tag\)](#)
- [BaseRecord \(const std::string& recordName, data\\_type\\_enum rt, Interface \\*puser=nullptr, Interface \\*pplc=nullptr\)](#)
- [virtual ~BaseRecord \(\)](#)  
*Destructor.*

- const `std::stringcase & get_name () const`  
*Get name.*
- void `set_name (const std::stringcase &recordName)`  
*Set name.*
- bool `get_process () const`  
*Get process flag: false = disabled, true = enabled.*
- void `set_process (bool isEnabled)`  
*Set process flag.*
- `access_rights_enum get_access_rights ()`  
*Get access rights.*
- void `set_access_rights (access_rights_enum rights)`  
*Set access rights.*
- `time_type get_timestamp () const`  
*Get time stamp.*
- virtual `Interface * get_userInterface () const`  
*Get pointer to user interface (no ownership transfer)*
- virtual `Interface * get_plcInterface () const`  
*Get pointer to plc interface (no ownership transfer)*
- void `set_userInterface (Interface *puser)`  
*Set user interface (object will be adopted!)*
- void `set_plcInterface (Interface *pplc)`  
*Set plc interface (object will be adopted!)*
- virtual `BasePLC * get_parent () const`  
*Get parent plc.*
- virtual void `set_parent (BasePLC *pPLC)`  
*Set parent plc.*
- const `DataValue & get_data () const`  
*Get a const reference to the data object.*
- `DataValue & get_data ()`  
*Get a reference to the data object.*
- bool `DatasValid ()`  
*Returns true, if the data is valid.*
- template<typename T >  
 bool `UserRead (T &data)`
- bool `UserRead (type_string_value *data, size_type max)`
- bool `UserRead (DataValue::type_wstring_value *data, size_type max)`
- template<typename T >  
 bool `UserWrite (const T &data)`
- bool `UserWrite (const type_string_value *data, size_type max)`
- bool `UserWrite (const type_wstring_value *data, size_type max)`
- `size_type UserReadBinary (type_binary p, size_type len)`
- `size_type UserWriteBinary (const type_binary p, size_type len)`
- bool `UserIsDirty () const`  
*Checks if the user needs to read an updated value.*
- void `UserSetDirty ()`  
*Set dirty flag for user.*
- bool `UserPush (bool force=false)`
- bool `UserPull ()`  
*Pulls the user for new data.*
- void `UserSetValid (bool valid)`
- bool `UserGetValid ()`

- template<typename T >  
bool **PlcRead** (T &data)
- bool **PlcRead** (type\_string\_value \*data, size\_type max)
- bool **PlcRead** (type\_wstring\_value \*data, size\_type max)
- template<typename T >  
bool **PlcWrite** (const T &data)
- bool **PlcWrite** (const type\_string\_value \*data, size\_type max)
- bool **PlcWrite** (const type\_wstring\_value \*data, size\_type max)
- size\_type **PlcReadBinary** (type\_binary p, size\_type len)
- size\_type **PlcWriteBinary** (const type\_binary p, size\_type len)
- bool **PlcIsDirty** () const  
*Checks if the plc needs to read an updated value.*
- void **PlcSetDirty** ()  
*Set dirty flag for plc.*
- bool **PlcPush** (bool force=false)
- bool **PlcPull** ()  
*Pulls the plc for new data.*
- void **PlcSetValid** (bool valid)
- bool **PlcGetValid** ()

## Protected Attributes

- std::stringcase name  
*Name.*
- access\_rights\_enum access  
*Enum for access rights.*
- atomic\_bool process  
*Process flag: false = disabled, true = enabled.*
- **DataValue** value  
*Data value.*
- InterfacePtr plc  
*PLC interface (master)*
- InterfacePtr user  
*User interface (slave)*
- BasePLC \* parent  
*PLC that this record belongs to.*

## Additional Inherited Members

### 8.8.1 Detailed Description

Class for managing a tag/channel.

This is the base class for a tag/channel. It contains a data value and owns two pointers to a user and plc interface, respectively. This is the basic work horse for exchanging data between a user (slave) and a plc (master).

Definition at line 495 of file plcBase.h.

## 8.8.2 Constructor & Destructor Documentation

### 8.8.2.1 BaseRecord() [1/2]

```
plc::BaseRecord::BaseRecord (
    const std::string& tag ) [inline], [explicit]
```

Constructor

Parameters

|            |                     |
|------------|---------------------|
| <i>tag</i> | Name of tag/channel |
|------------|---------------------|

Definition at line 502 of file plcBase.h.

References plc::read\_write.

### 8.8.2.2 BaseRecord() [2/2]

```
plc::BaseRecord::BaseRecord (
    const std::string& recordName,
    data_type_enum rt,
    Interface * puser = nullptr,
    Interface * pplc = nullptr ) [inline]
```

Constructor

Parameters

|                   |   |
|-------------------|---|
| <i>recordName</i> | Name of tag/channel                                 |
| <i>rt</i>         | Data type   |
| <i>puser</i>      | Pointer to user interface object (will be adopted!) |
| <i>pplc</i>       | Pointer to plc interface object (will be adopted!)  |

Definition at line 509 of file plcBase.h.

References plc::read\_write.

## 8.8.3 Member Function Documentation

### 8.8.3.1 PlcGetValid()

```
bool plc::BaseRecord::PlcGetValid () [inline]
```

Get the plc valid flag and reset the dirty flag

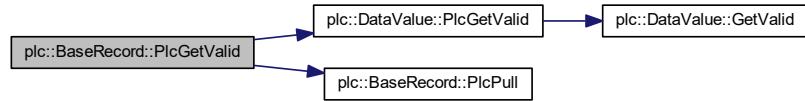
#### Returns

valid True for valid data, False for invalid

Definition at line 682 of file plcBase.h.

References plc::DataValue::PlcGetValid(), PlcPull(), process, and value.

Here is the call graph for this function:



### 8.8.3.2 PlcPush()

```
bool plc::BaseRecord::PlcPush (
    bool force = false ) [inline]
```

Initiated a plc pull, if the data needs an update

#### Parameters

|                    |  |
|--------------------|--|
| <code>force</code> | Forces a plc update even when not needed |
|--------------------|--|

Definition at line 179 of file plcBaseTemplate.h.

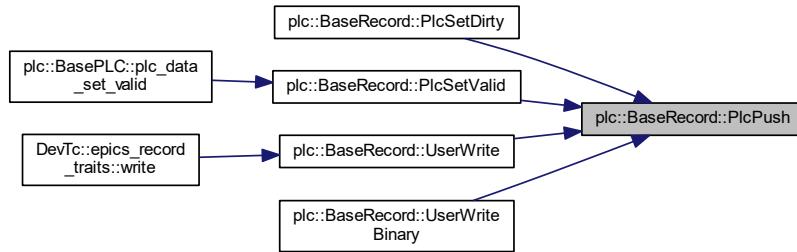
References PlcIsDirty().

Referenced by PlcSetDirty(), PlcSetValid(), UserWrite(), and UserWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.8.3.3 PlcRead() [1/3]

```
template<typename T >
bool plc::BaseRecord::PlcRead (
    T & data ) [inline]
```

Execute a plc read, but pull user first

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>data</code> | Reference to data (return) |
|-------------------|----------------------------|

#### Returns

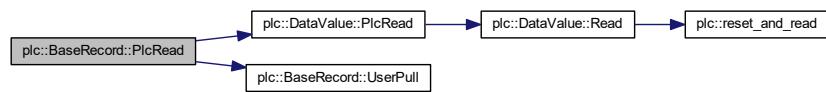
true if successfull

Definition at line 621 of file `plcBase.h`.

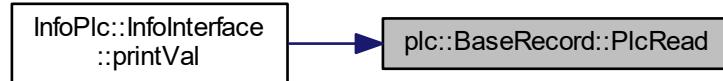
References `plc::DataValue::PlcRead()`, `UserPull()`, and `value`.

Referenced by `InfoPlc::InfoInterface::printVal()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.8.3.4 PlcRead() [2/3]

```
bool plc::BaseRecord::PlcRead (
    type_string_value * data,
    size_type max ) [inline]
```

Execute a plc read, but pull user first

##### Parameters

|             |                                   |
|-------------|-----------------------------------|
| <i>data</i> | character pointer, pchar (return) |
| <i>max</i>  | Maximum number of characters      |

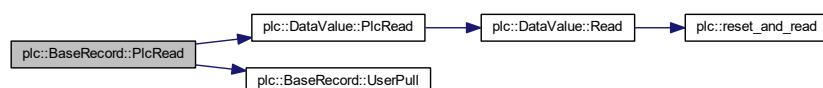
##### Returns

true if successfull

Definition at line 627 of file plcBase.h.

References plc::DataValue::PlcRead(), UserPull(), and value.

Here is the call graph for this function:



#### 8.8.3.5 PlcRead() [3/3]

```
bool plc::BaseRecord::PlcRead (
    type_wstring_value * data,
    size_type max ) [inline]
```

Execute a plc read, but pull user first

**Parameters**

|             |   |
|-------------|---|
| <i>data</i> | character pointer, <code>pwchar</code> (return) |
| <i>max</i>  | Maximum number of characters                    |

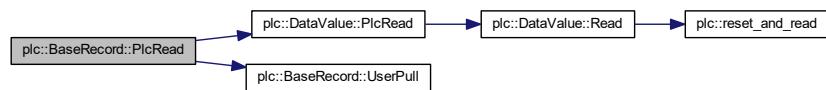
**Returns**

true if successfull

Definition at line 633 of file `plcBase.h`.

References `plc::DataValue::PlcRead()`, `UserPull()`, and `value`.

Here is the call graph for this function:

**8.8.3.6 PlcReadBinary()**

```
size_type plc::BaseRecord::PlcReadBinary (
    type_binary p,
    size_type len ) [inline]
```

Execute a plc read, but pull user first

**Parameters**

|            |   |
|------------|---|
| <i>p</i>   | Pointer to data (destination buffer)              |
| <i>len</i> | Length in bytes (must be the same as data length) |

**Returns**

Number of bytes read (0 on error)

Definition at line 658 of file `plcBase.h`.

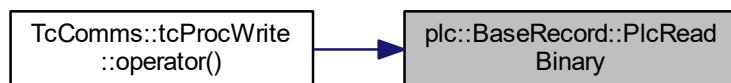
References `plc::DataValue::PlcReadBinary()`, `UserPull()`, and `value`.

Referenced by `TcComms::tcProcWrite::operator()()`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.8.3.7 PlcSetValid()

```
void plc::BaseRecord::PlcSetValid (
    bool valid ) [inline]
```

Set the plc valid flag and set the dirty flag when flag changes

#### Parameters

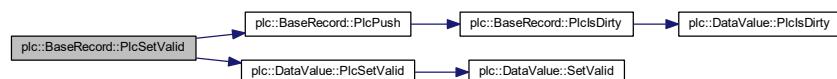
|                    |  |
|--------------------|--|
| <code>valid</code> | True for valid data, False for invalid |
|--------------------|--|

Definition at line 679 of file plcBase.h.

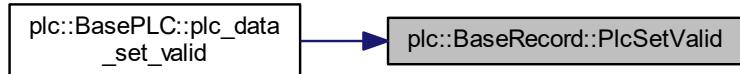
References PlcPush(), plc::DataValue::PlcSetValid(), and value.

Referenced by plc::BasePLC::plc\_data\_set\_valid().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.8.3.8 PlcWrite() [1/3]

```
template<typename T >
bool plc::BaseRecord::PlcWrite (
    const T & data ) [inline]
```

Execute a plc write and push user

#### Parameters

|             |                   |
|-------------|-------------------|
| <i>data</i> | Reference to data |
|-------------|-------------------|

#### Returns

true if successfull

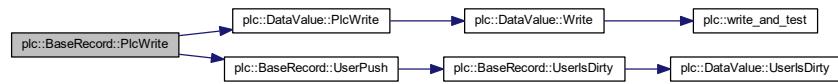
Definition at line 639 of file plcBase.h.

References `plc::DataValue::PlcWrite()`, `UserPush()`, and `value`.

Referenced by `InfoPlc::InfoInterface::info_update_active()`, `InfoPlc::InfoInterface::info_update_ads_build()`, `InfoPlc::InfoInterface::info_update_ads_netid_b0()`, `InfoPlc::InfoInterface::info_update_ads_netid_b1()`, `InfoPlc::InfoInterface::info_update_ads_netid_b2()`, `InfoPlc::InfoInterface::info_update_ads_netid_b3()`, `InfoPlc::InfoInterface::info_update_ads_netid_b4()`, `InfoPlc::InfoInterface::info_update_ads_netid_b5()`, `InfoPlc::InfoInterface::info_update_ads_netid_str()`, `InfoPlc::InfoInterface::info_update_ads_port()`, `InfoPlc::InfoInterface::info_update_ads_revision()`, `InfoPlc::InfoInterface::info_update_ads_version()`, `InfoPlc::InfoInterface::info_update_alias()`, `InfoPlc::InfoInterface::info_update_callback_queue0_free()`, `InfoPlc::InfoInterface::info_update_callback_queue0_percent()`, `InfoPlc::InfoInterface::info_update_callback_queue0_size()`, `InfoPlc::InfoInterface::info_update_callback_queue0_used()`, `InfoPlc::InfoInterface::info_update_callback_queue1_free()`, `InfoPlc::InfoInterface::info_update_callback_queue1_percent()`, `InfoPlc::InfoInterface::info_update_callback_queue1_size()`, `InfoPlc::InfoInterface::info_update_callback_queue1_used()`, `InfoPlc::InfoInterface::info_update_callback_queue2_free()`, `InfoPlc::InfoInterface::info_update_callback_queue2_percent()`, `InfoPlc::InfoInterface::info_update_callback_queue2_size()`, `InfoPlc::InfoInterface::info_update_callback_queue2_used()`, `InfoPlc::InfoInterface::info_update_name()`, `InfoPlc::InfoInterface::info_update_rate_read()`, `InfoPlc::InfoInterface::info_update_rate_update()`, `InfoPlc::InfoInterface::info_update_rate_write()`, `InfoPlc::InfoInterface::info_update_records_num()`, `InfoPlc::InfoInterface::info_update_state()`, `InfoPlc::InfoInterface::info_update_statestr()`, `InfoPlc::InfoInterface::info_update_svn_local()`, `InfoPlc::InfoInterface::info_update_svn_revision()`, `InfoPlc::InfoInterface::info_update_svn_time()`, `InfoPlc::InfoInterface::info_update_timestamp_day()`, `InfoPlc::InfoInterface::info_update_timestamp_hour()`, `InfoPlc::InfoInterface::info_update_timestamp_min()`, `InfoPlc::InfoInterface::info_update_timestamp_month()`, `InfoPlc::InfoInterface::info_update_timestamp_sec()`, `InfoPlc::InfoInterface::info_update_timestamp_str()`, `InfoPlc::InfoInterface::info_update_timestamp_year()`, `InfoPlc::InfoInterface::info_update_tpy_filename()`, `InfoPlc::InfoInterface::info_update_tpy_time_day()`, `InfoPlc::InfoInterface::info_update_tpy_time_hour()`, `InfoPlc::InfoInterface::info_update_tpy_time_min()`,

InfoPlc::InfoInterface::info\_update\_tpy\_time\_month(), InfoPlc::InfoInterface::info\_update\_tpy\_time\_sec(), InfoPlc::InfoInterface::info\_update\_tpy\_time\_str(), InfoPlc::InfoInterface::info\_update\_tpy\_time\_year(), and InfoPlc::InfoInterface::info\_update\_tpy\_valid().

Here is the call graph for this function:



### 8.8.3.9 PlcWrite() [2/3]

```
bool plc::BaseRecord::PlcWrite (
    const type_string_value * data,
    size_type max ) [inline]
```

Execute a plc write and push user

#### Parameters

|             |                              |
|-------------|------------------------------|
| <i>data</i> | character pointer, pchar     |
| <i>max</i>  | Maximum number of characters |

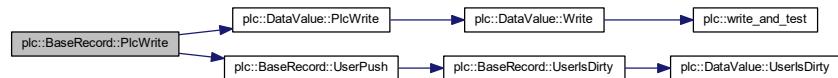
#### Returns

true if successfull

Definition at line 645 of file plcBase.h.

References plc::DataValue::PlcWrite(), UserPush(), and value.

Here is the call graph for this function:



### 8.8.3.10 PlcWrite() [3/3]

```
bool plc::BaseRecord::PlcWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Execute a plc write and push user

**Parameters**

|             |  |
|-------------|--|
| <i>data</i> | character pointer, <code>pwchar</code> |
| <i>max</i>  | Maximum number of characters           |

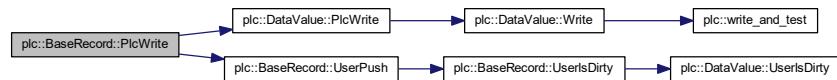
**Returns**

true if successfull

Definition at line 651 of file `plcBase.h`.

References `plc::DataValue::PlcWrite()`, `UserPush()`, and `value`.

Here is the call graph for this function:

**8.8.3.11 PlcWriteBinary()**

```
size_type plc::BaseRecord::PlcWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Execute a plc write and push user

**Parameters**

|            |   |
|------------|---|
| <i>p</i>   | Pointer to data (source buffer)                   |
| <i>len</i> | Length in bytes (must be the same as data length) |

**Returns**

Number of bytes written (0 on error)

Definition at line 664 of file `plcBase.h`.

References `plc::DataValue::PlcWriteBinary()`, `UserPush()`, and `value`.

Referenced by `TcComms::TcPLC::read_scanner()`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.8.3.12 UserGetValid()

```
bool plc::BaseRecord::UserGetValid() [inline]
```

Get the user valid flag and reset the dirty flag

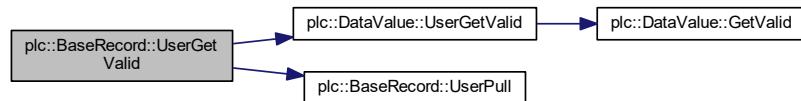
#### Returns

valid True for valid data, False for invalid

Definition at line 615 of file plcBase.h.

References process, plc::DataValue::UserGetValid(), UserPull(), and value.

Here is the call graph for this function:



### 8.8.3.13 UserPush()

```
bool plc::BaseRecord::UserPush(
    bool force = false) [inline]
```

Initiated a user pull, if the data needs an update

**Parameters**

|              |   |
|--------------|---|
| <i>force</i> | Forces a user update even when not needed |
|--------------|---|

Definition at line 153 of file plcBaseTemplate.h.

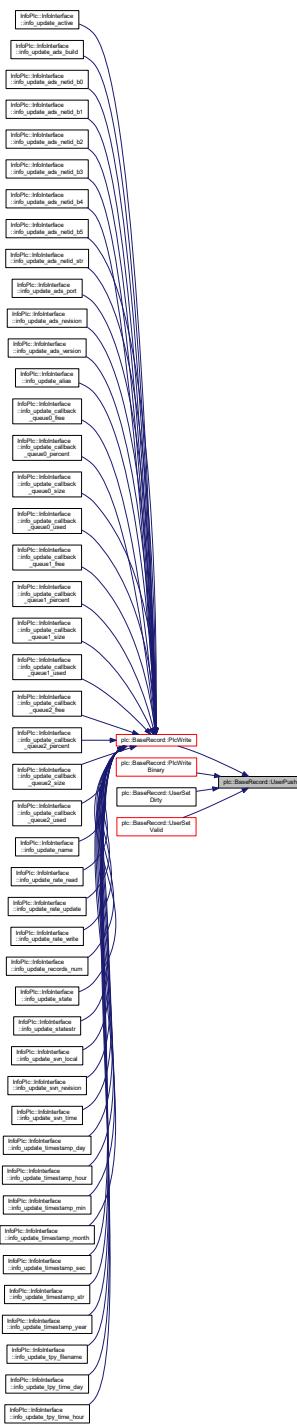
References user, and UserIsDirty().

Referenced by PlcWrite(), PlcWriteBinary(), UserSetDirty(), and UserSetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.8.3.14 UserRead() [1/3]

```

bool plc::BaseRecord::UserRead (
    DataValue::type_wstring_value * data,
    size_type max ) [inline]

```

Execute a user read, but pull plc first

#### Parameters

|             |                                    |
|-------------|------------------------------------|
| <i>data</i> | character pointer, pwchar (return) |
| <i>max</i>  | Maximum number of characters       |

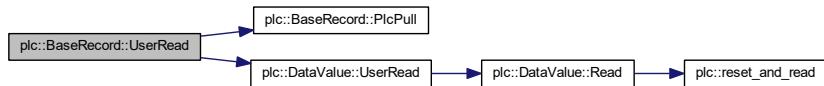
#### Returns

true if successfull

Definition at line 567 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

Here is the call graph for this function:



### 8.8.3.15 UserRead() [2/3]

```
template<typename T >
bool plc::BaseRecord::UserRead (
    T & data )  [inline]
```

Execute a user read, but pull plc first

#### Parameters

|             |                            |
|-------------|----------------------------|
| <i>data</i> | Reference to data (return) |
|-------------|----------------------------|

#### Returns

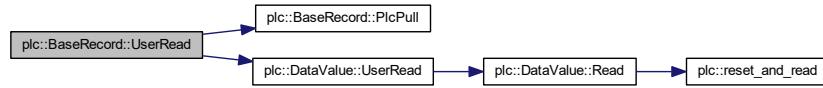
true if successfull

Definition at line 555 of file plcBase.h.

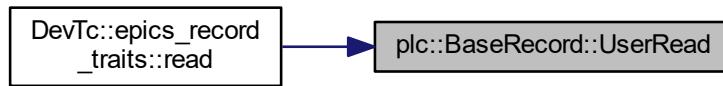
References PlcPull(), plc::DataValue::UserRead(), and value.

Referenced by DevTc::epics\_record\_traits< RecType >::read().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.8.3.16 UserRead() [3/3]

```
bool plc::BaseRecord::UserRead (
    type_string_value * data,
    size_type max ) [inline]
```

Execute a user read, but pull plc first

#### Parameters

|             |                                   |
|-------------|-----------------------------------|
| <i>data</i> | character pointer, pchar (return) |
| <i>max</i>  | Maximum number of characters      |

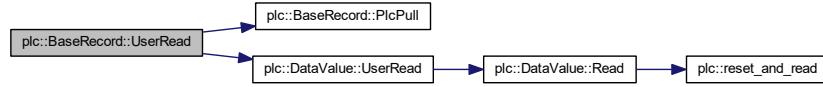
#### Returns

true if successfull

Definition at line 561 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

Here is the call graph for this function:



### 8.8.3.17 UserReadBinary()

```
size_type plc::BaseRecord::UserReadBinary (
    type_binary p,
    size_type len ) [inline]
```

Execute a user read, but pull plc first

#### Parameters

|            |   |
|------------|---|
| <i>p</i>   | Pointer to data (destination buffer)              |
| <i>len</i> | Length in bytes (must be the same as data length) |

#### Returns

Number of bytes read (0 on error)

Definition at line 591 of file plcBase.h.

References PlcPull(), plc::DataValue::UserReadBinary(), and value.

Here is the call graph for this function:



### 8.8.3.18 UserSetValid()

```
void plc::BaseRecord::UserSetValid (
    bool valid ) [inline]
```

Set the user valid flag and set the dirty flag when flag changes

#### Parameters

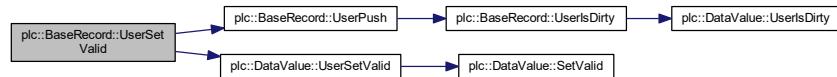
|              |  |
|--------------|--|
| <i>valid</i> | True for valid data, False for invalid |
|--------------|--|

Definition at line 612 of file plcBase.h.

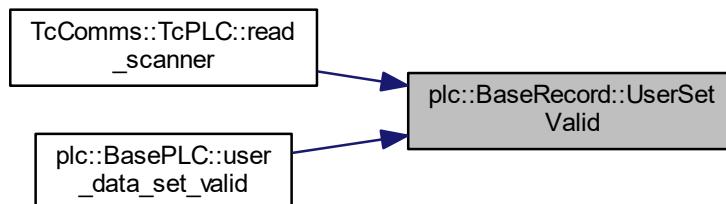
References UserPush(), plc::DataValue::UserSetValid(), and value.

Referenced by TcComms::TcPLC::read\_scanner(), and plc::BasePLC::user\_data\_set\_valid().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.8.3.19 UserWrite() [1/3]

```
template<typename T >
bool plc::BaseRecord::UserWrite (
    const T & data ) [inline]
```

Execute a user write and push plc

#### Parameters

|             |                   |
|-------------|-------------------|
| <i>data</i> | Reference to data |
|-------------|-------------------|

#### Returns

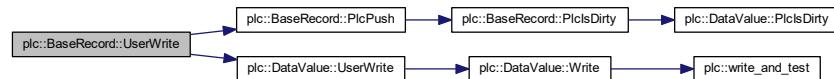
true if successfull

Definition at line 572 of file plcBase.h.

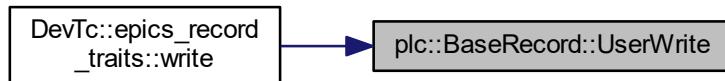
References PlcPush(), plc::DataValue::UserWrite(), and value.

Referenced by DevTc::epics\_record\_traits< RecType >::write().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.8.3.20 UserWrite() [2/3]

```

bool plc::BaseRecord::UserWrite (
    const type_string_value * data,
    size_type max ) [inline]
    
```

Execute a user write and push plc

#### Parameters

|             |                              |
|-------------|------------------------------|
| <i>data</i> | character pointer, pchar     |
| <i>max</i>  | Maximum number of characters |

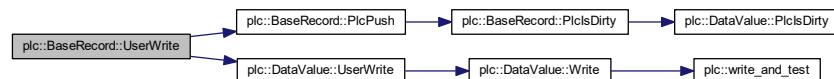
#### Returns

true if successfull

Definition at line 578 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

Here is the call graph for this function:



### 8.8.3.21 UserWrite() [3/3]

```
bool plc::BaseRecord::UserWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Execute a user write and push plc

#### Parameters

|             |                              |
|-------------|------------------------------|
| <i>data</i> | character pointer, pwchar    |
| <i>max</i>  | Maximum number of characters |

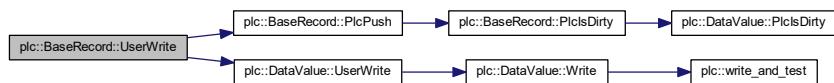
#### Returns

true if successfull

Definition at line 584 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

Here is the call graph for this function:



### 8.8.3.22 UserWriteBinary()

```
size_type plc::BaseRecord::UserWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Execute a user write and push plc

#### Parameters

|            |   |
|------------|---|
| <i>p</i>   | Pointer to data (source buffer)                   |
| <i>len</i> | Length in bytes (must be the same as data length) |

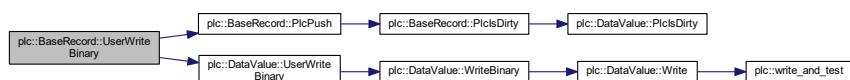
**Returns**

Number of bytes written (0 on error)

Definition at line 597 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWriteBinary(), and value.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

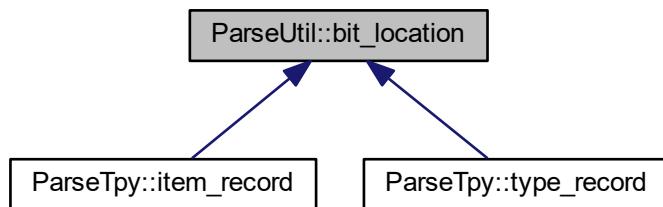
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

## 8.9 ParseUtil::bit\_location Class Reference

Bit location.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::bit\_location:



## Public Member Functions

- [bit\\_location \(\)](#)  
*Default constructor.*
- [bit\\_location \(int bo, int bs\)](#)  
*Constructor.*
- [bool isValid \(\) const](#)  
*Validity.*
- [const int get\\_bit\\_offset \(\) const](#)  
*Get bit offset.*
- [void set\\_bit\\_offset \(int ofs\)](#)  
*Set bit offset.*
- [const int get\\_bit\\_size \(\) const](#)  
*Get bit size.*
- [void set\\_bit\\_size \(int size\)](#)  
*Set bit size.*

## Protected Attributes

- [int bitoffs](#)  
*bit offset where elements is stored*
- [int bitsize](#)  
*size in number of bits of symbol*

### 8.9.1 Detailed Description

Bit location.

This is a class for storing bit offset and size in a structure

Definition at line 233 of file ParseUtil.h.

The documentation for this class was generated from the following file:

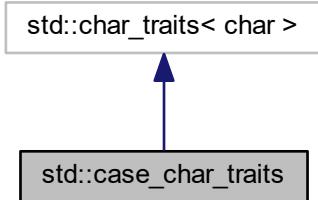
- [ParseUtil.h](#)

## 8.10 std::case\_char\_traits Struct Reference

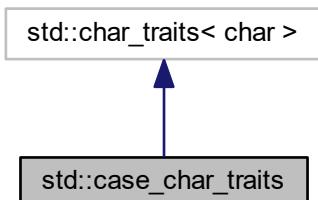
case insensitive traits.

```
#include <stringcase.h>
```

Inheritance diagram for std::case\_char\_traits:



Collaboration diagram for std::case\_char\_traits:



## Static Public Member Functions

- static bool [eq](#) (const char\_type &c1, const char\_type &c2)
- static bool [ne](#) (const char\_type &c1, const char\_type &c2)
- static bool [lt](#) (const char\_type &c1, const char\_type &c2)
- static int [compare](#) (const char\_type \*s1, const char\_type \*s2, size\_t n)

### 8.10.1 Detailed Description

case insensitive traits.

This traits class is not case sensitive.

Definition at line 40 of file stringcase.h.

The documentation for this struct was generated from the following file:

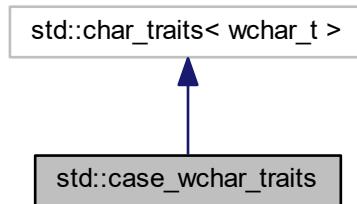
- [stringcase.h](#)

## 8.11 std::case\_wchar\_traits Struct Reference

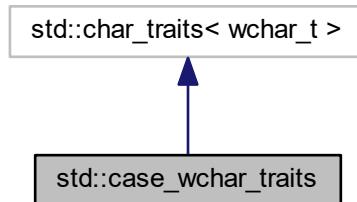
case insensitive unicode traits.

```
#include <stringcase.h>
```

Inheritance diagram for std::case\_wchar\_traits:



Collaboration diagram for std::case\_wchar\_traits:



### Static Public Member Functions

- static bool [eq](#) (const char\_type &c1, const char\_type &c2)
- static bool [ne](#) (const char\_type &c1, const char\_type &c2)
- static bool [lt](#) (const char\_type &c1, const char\_type &c2)
- static int [compare](#) (const char\_type \*s1, const char\_type \*s2, size\_t n)

#### 8.11.1 Detailed Description

case insensitive unicode traits.

This unicode traits class is not case sensitive.

Definition at line 72 of file stringcase.h.

The documentation for this struct was generated from the following file:

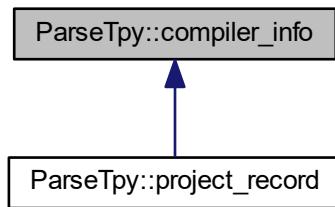
- [stringcase.h](#)

## 8.12 ParseTpy::compiler\_info Class Reference

Compiler information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::compiler\_info:



### Public Member Functions

- `compiler_info ()`  
*Default constructor.*
- `const std::stringcase & get_cmpl_versionstr () const`  
*Get compiler version string.*
- `void set_cmpl_versionstr (const std::stringcase &versionstr)`  
*Set compiler version string.*
- `double get_cmpl_version () const`  
*Get compiler version.*
- `const std::stringcase & get_tcat_versionstr () const`  
*Get twincat version string.*
- `void set_tcat_versionstr (const std::stringcase &versionstr)`  
*Set twincat version string.*
- `unsigned int get_tcat_version_major () const`  
*Get twincat major version.*
- `unsigned int get_tcat_version_minor () const`  
*Get twincat minor version.*
- `unsigned int get_tcat_version_build () const`  
*Get twincat build version.*
- `const std::stringcase & get_cpu_family () const`  
*Get cpu family string.*
- `void set_cpu_family (const std::stringcase &family)`  
*Set cpu family string.*
- `bool is_cmpl_Valid () const`  
*Checks, if version is of the form n.n...*
- `bool is_tcat_Valid () const`  
*Checks, if twincat version is of the form n.n...*

## Protected Attributes

- `std::string` `case cmpl_versionstr`  
*version string*
- `double` `cmpl_version`  
*version number*
- `std::string` `case tcat_versionstr`  
*twincat version string*
- `unsigned int` `tcatalog_version_major`  
*twincat major version number*
- `unsigned int` `tcatalog_version_minor`  
*twincat minor version number*
- `unsigned int` `tcatalog_version_build`  
*twincat build version number*
- `std::string` `cpu_family`  
*cpu family string*

### 8.12.1 Detailed Description

Compiler information.

This is a base class for storing the compiler information

Definition at line 80 of file ParseTpy.h.

The documentation for this class was generated from the following files:

- `ParseTpy.h`
- `ParseTpy.cpp`

## 8.13 TcComms::DataPar Struct Reference

Memory location struct.

```
#include <tcComms.h>
```

## Public Attributes

- `unsigned long` `indexGroup`  
*index group in ADS server*
- `unsigned long` `indexOffset`  
*index offset in ADS server*
- `unsigned long` `length`  
*count of bytes to read*

### 8.13.1 Detailed Description

Memory location struct.

Struct for storing index group, index offset, and size of a TC symbol

Definition at line 57 of file tcComms.h.

The documentation for this struct was generated from the following file:

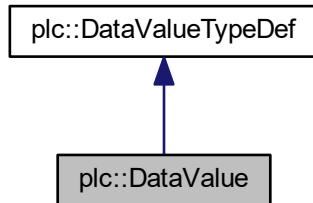
- [tcComms.h](#)

## 8.14 plc::DataValue Class Reference

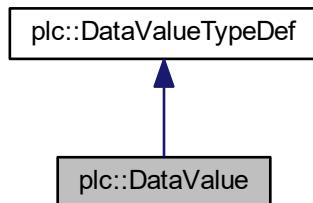
Data value.

```
#include <plcBase.h>
```

Inheritance diagram for plc::DataValue:



Collaboration diagram for plc::DataValue:



## Public Types

- `typedef void * data_type`  
*Internally used storage pointer type.*

## Public Member Functions

- `DataValue ()`  
*Default constructor.*
- `DataValue (data_type_enum rt, size_type len=0)`
- `~DataValue ()`  
*Destructor.*
- `DataValue (const DataValue &)`  
*Copy constructor.*
- `DataValue & operator= (const DataValue &)`  
*Assignment operator.*
- `void Init (data_type_enum rt, size_type len=0)`
- `bool IsValid () const`  
*is valid*
- `data_type_enum get_data_type () const`  
*get type*
- `size_type get_size () const`  
*get size*
- template<typename T >  
  `bool UserRead (T &data) const`
- template<size\_type N>  
  `bool UserRead (type_string_value(&data)[N]) const`
- `bool UserRead (type_string_value *data, size_type max) const`
- `bool UserRead (type_wstring_value *data, size_type max) const`
- template<typename T >  
  `bool UserWrite (const T &data)`
- template<size\_type N>  
  `bool UserWrite (const type_string_value(&data)[N])`
- `bool UserWrite (const type_string_value *data, size_type max)`
- `bool UserWrite (const type_wstring_value *data, size_type max)`
- `size_type UserReadBinary (type_binary p, size_type len) const`
- `size_type UserWriteBinary (const type_binary p, size_type len)`
- `bool UserIsDirty () const`  
*New data for user.*
- `void UserSetDirty ()`  
*Set dirty flag for user.*
- `void UserSetValid (bool valid)`
- `bool UserGetValid () const`
- template<typename T >  
  `bool PlcRead (T &data) const`
- template<size\_type N>  
  `bool PlcRead (type_string_value(&data)[N]) const`
- `bool PlcRead (type_string_value *data, size_type max) const`
- `bool PlcRead (type_wstring_value *data, size_type max) const`
- template<typename T >  
  `bool PlcWrite (const T &data)`
- template<size\_type N>  
  `bool PlcWrite (const type_string_value(&data)[N])`

- `bool PlcWrite (const type_string_value *data, size_type max)`
- `bool PlcWrite (const type_wstring_value *data, size_type max)`
- `size_type PlcReadBinary (type_binary p, size_type len) const`
- `size_type PlcWriteBinary (const type_binary p, size_type len)`
- `bool PlcIsDirty () const`  
*New data for plc.*
- `void PlcSetDirty ()`  
*Set dirty flag for plc.*
- `void PlcSetValid (bool valid)`
- `bool PlcGetValid () const`

## Protected Member Functions

- `DataValue (const DataValue &&)`  
*Constructor (hidden)*
- `DataValue & operator= (const DataValue &&)`  
*Assignment operator.*
- `template<typename T >`  
`bool Read (atomic_bool &dirty, T &data) const`
- `bool Read (atomic_bool &dirty, type_string &data) const`
- `bool Read (atomic_bool &dirty, type_wstring &data) const`
- `bool Read (atomic_bool &dirty, type_string_value *data, size_type max) const`
- `bool Read (atomic_bool &dirty, type_wstring_value *data, size_type max) const`
- `template<typename T >`  
`bool Write (atomic_bool &dirty, const atomic_bool &pend, const T &data)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_string &data)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_wstring &data)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_string_value *data, size_type max)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_wstring_value *data, size_type max)`
- `size_type ReadBinary (atomic_bool &dirty, type_binary p, size_type len) const`
- `size_type WriteBinary (atomic_bool &dirty, const atomic_bool &pend, const type_binary p, size_type len)`
- `void SetValid (atomic_bool &dirty, bool valid)`
- `bool GetValid (atomic_bool &dirty) const`

## Protected Attributes

- `data_type mydata`  
*Data pointer.*
- `size_type mysize`
- `data_type_enum mytype`  
*Data type.*
- `atomic_bool myvalid`  
*Valid flag.*
- `atomic_bool myuserdirty`  
*Dirty flag indicating user needs to update.*
- `atomic_bool myplcdirty`  
*Dirty flag indicating plc needs to update.*

## Additional Inherited Members

### 8.14.1 Detailed Description

Data value.

Class for data value This class stores a data value and provides synchronization between the user (slave) and the plc (master) interfaces. When the plc writes a value, it is marked dirty on the user side. Then, when the user reads this data, it resets the dirty flag. The same logic applies for writes by the user and reads by the plc.

Data access is guaranteed to be atomic and MT safe. Construction, initialization and destruction is not MT safe and all data access has to be stopped during these operations.

Type conversion is provided between all simple data types. However, the loss of information is not checked upon a down cast. Strings have to be read as strings. Binary data needs to be accessed with the binary read/write operations. However, all data can be accessed through binary access.

Definition at line 223 of file plcBase.h.

### 8.14.2 Constructor & Destructor Documentation

#### 8.14.2.1 DataValue()

```
plc::DataValue::DataValue (
    data_type_enum rt,
    size_type len = 0 ) [inline], [explicit]
```

Constructor

Parameters

|            |                             |
|------------|-----------------------------|
| <i>rt</i>  | Data type enumeration value |
| <i>len</i> | Length of data              |

Definition at line 235 of file plcBase.h.

References plc::dtlInvalid, and Init().

Here is the call graph for this function:



## 8.14.3 Member Function Documentation

### 8.14.3.1 GetValid()

```
bool plc::DataValue::GetValid (
    atomic_bool & dirty ) const [protected]
```

Get the valid flag and reset the dirty flag

#### Parameters

|              |                                       |
|--------------|---------------------------------------|
| <i>dirty</i> | Reference to dirty flag (user or plc) |
|--------------|---------------------------------------|

#### Returns

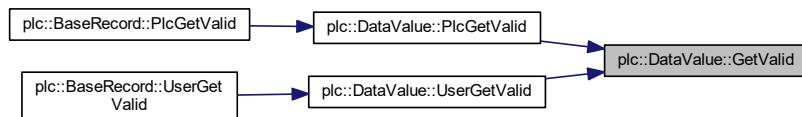
valid True for valid data, False for invalid

Definition at line 508 of file plcBase.cpp.

References plc::DataValueTypeDef::memory\_order, and myvalid.

Referenced by PlcGetValid(), and UserGetValid().

Here is the caller graph for this function:



### 8.14.3.2 Init()

```
void plc::DataValue::Init (
    data_type_enum rt,
    size_type len = 0 )
```

Initializes data value

#### Parameters

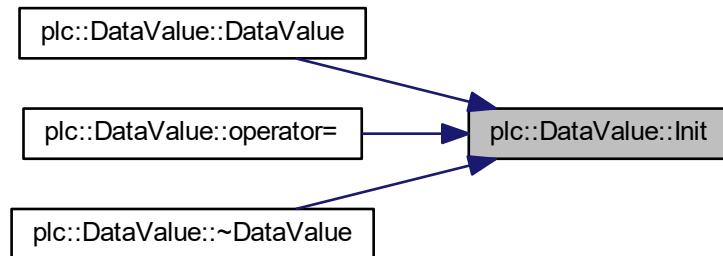
|            |                                      |
|------------|--------------------------------------|
| <i>rt</i>  | Data type enumeration value          |
| <i>len</i> | Length of data (use only for binary) |

Definition at line 187 of file plcBase.cpp.

References `plc::dtBinary`, `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtInvalid`, `plc::dtString`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `plc::dtWString`, `mydata`, `myplcdirty`, `mysize`, `mytype`, and `myuserdirty`.

Referenced by `DataValue()`, `operator=()`, and `~DataValue()`.

Here is the caller graph for this function:



#### 8.14.3.3 PlcGetValid()

```
bool plc::DataValue::PlcGetValid( ) const [inline]
```

Get the valid flag and reset the dirty flag

Returns

`valid` True for valid data, False for invalid

Definition at line 375 of file plcBase.h.

References `GetValid()`, and `myuserdirty`.

Referenced by `plc::BaseRecord::PlcGetValid()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.14.3.4 PlcRead() [1/4]

```
template<typename T >
bool plc::DataValue::PlcRead (
    T & data ) const [inline]
```

Read data by the plc

##### Parameters

|             |                               |
|-------------|-------------------------------|
| <i>data</i> | Data value reference (return) |
|-------------|-------------------------------|

Definition at line 320 of file plcBase.h.

References myplcdirty, and Read().

Referenced by plc::BaseRecord::PlcRead().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.14.3.5 PlcRead() [2/4]

```
bool plc::DataValue::PlcRead (
    type_string_value * data,
    size_type max ) const [inline]
```

Read character array (pchar) by the plc

#### Parameters

|             |                    |
|-------------|--------------------|
| <i>data</i> | Destination buffer |
| <i>max</i>  | Maximum length     |

Definition at line 329 of file plcBase.h.

References myplcdirty, and Read().

Here is the call graph for this function:



### 8.14.3.6 PlcRead() [3/4]

```
template<size_type N>
bool plc::DataValue::PlcRead (
    type_string_value(&) data[N] ) const [inline]
```

Read fixed length character array data by the plc

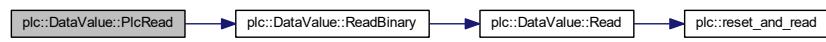
#### Parameters

|             |  |
|-------------|--|
| <i>data</i> | Data value reference for a fixed length character array (return) |
|-------------|--|

Definition at line 324 of file plcBase.h.

References myplcdirty, and ReadBinary().

Here is the call graph for this function:



### 8.14.3.7 PlcRead() [4/4]

```
bool plc::DataValue::PlcRead (
    type_wstring_value * data,
    size_type max ) const [inline]
```

Read character array (pwchar) by the plc

#### Parameters

|             |                    |
|-------------|--------------------|
| <i>data</i> | Destination buffer |
| <i>max</i>  | Maximum length     |

Definition at line 334 of file plcBase.h.

References myplcdirty, and Read().

Here is the call graph for this function:



### 8.14.3.8 PlcReadBinary()

```
size_type plc::DataValue::PlcReadBinary (
    type_binary p,
    size_type len ) const [inline]
```

Read data as binary by the plc

#### Parameters

|            |                                    |
|------------|------------------------------------|
| <i>p</i>   | value pointer (destination buffer) |
| <i>len</i> | Length in bytes                    |

Definition at line 358 of file plcBase.h.

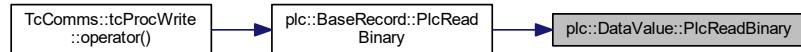
References myplcdirty, and ReadBinary().

Referenced by plc::BaseRecord::PlcReadBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.14.3.9 PlcSetValid()

```
void plc::DataValue::PlcSetValid (  
    bool valid ) [inline]
```

Set the valid flag and set the dirty flag when flag changes

#### Parameters

|       |  |
|-------|--|
| valid | True for valid data, False for invalid |
|-------|--|

Definition at line 372 of file plcBase.h.

References myplcdirty, and SetValid().

Referenced by plc::BaseRecord::PlcSetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.14.3.10 PlcWrite() [1/4]

```
template<typename T >
bool plc::DataValue::PlcWrite (
    const T & data ) [inline]
```

Write data by the plc

##### Parameters

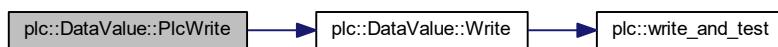
|             |                      |
|-------------|----------------------|
| <i>data</i> | Data value reference |
|-------------|----------------------|

Definition at line 338 of file plcBase.h.

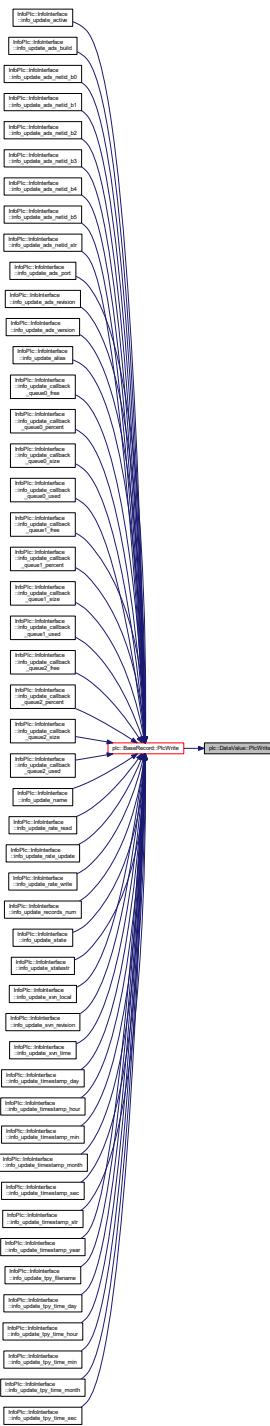
References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::PlcWrite().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.14.3.11 PlcWrite() [2/4]

```
bool plc::DataValue::PlcWrite (
    const type_string_value * data,
    size_type max ) [inline]
```

Write character array (pchar) by the plc

#### Parameters

|             |                |
|-------------|----------------|
| <i>data</i> | Source buffer  |
| <i>max</i>  | Maximum length |

Definition at line 347 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:



#### 8.14.3.12 PlcWrite() [3/4]

```
template<size_type N>
bool plc::DataValue::PlcWrite (
    const type_string_value(&) data[N] ) [inline]
```

Write fixed length character array data by the plc

#### Parameters

|             |                      |
|-------------|----------------------|
| <i>data</i> | Data value reference |
|-------------|----------------------|

Definition at line 342 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Here is the call graph for this function:



### 8.14.3.13 PlcWrite() [4/4]

```
bool plc::DataValue::PlcWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Write character array (wpchar) by the plc

#### Parameters

|             |                |
|-------------|----------------|
| <i>data</i> | Source buffer  |
| <i>max</i>  | Maximum length |

Definition at line 352 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:



### 8.14.3.14 PlcWriteBinary()

```
size_type plc::DataValue::PlcWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Write data as binary by the plc

#### Parameters

|            |                               |
|------------|-------------------------------|
| <i>p</i>   | value pointer (source buffer) |
| <i>len</i> | Length in bytes               |

Definition at line 363 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Referenced by plc::BaseRecord::PlcWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.14.3.15 Read() [1/5]

```
template<typename T >
bool plc::DataValue::Read (
    atomic_bool & dirty,
    T & data ) const [protected]
```

Read data

##### Parameters

|              |                                       |
|--------------|---------------------------------------|
| <i>dirty</i> | Reference to dirty flag (user or plc) |
| <i>data</i>  | Data value reference (return)         |

[DataValue::Read](#) (bool, Integral and floating point types)

Definition at line 87 of file plcBaseTemplate.h.

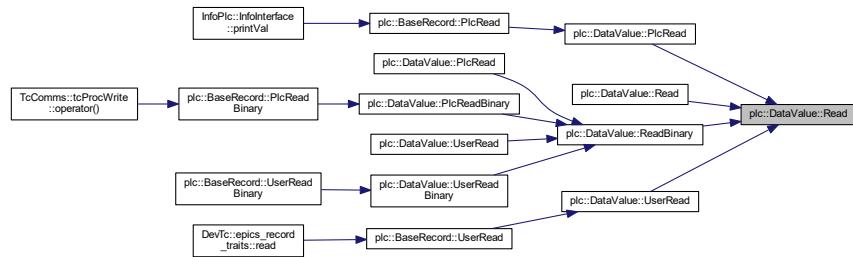
References `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `mydata`, `mytype`, and `plc::reset_and_read()`.

Referenced by `PlcRead()`, `Read()`, `ReadBinary()`, and `UserRead()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.14.3.16 Read() [2/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_string & data ) const [protected]
```

Read string (template specialization)

##### Parameters

|              |                                       |
|--------------|---------------------------------------|
| <i>dirty</i> | Reference to dirty flag (user or plc) |
| <i>data</i>  | Data value reference (return)         |

Definition at line 285 of file plcBase.cpp.

References plc::dtString, mydata, mytype, and plc::reset\_and\_read().

Here is the call graph for this function:



#### 8.14.3.17 Read() [3/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
```

```
type_string_value * data,  
size_type max ) const [protected]
```

Read character array (pchar)

**Parameters**

|              |                                       |
|--------------|---------------------------------------|
| <i>dirty</i> | Reference to dirty flag (user or plc) |
| <i>data</i>  | Destination buffer                    |
| <i>max</i>   | Maximum length                        |

Definition at line 312 of file plcBase.cpp.

References Read().

Here is the call graph for this function:

**8.14.3.18 Read() [4/5]**

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_wstring & data ) const [protected]
```

Read wstring (template specialization)

**Parameters**

|              |                                       |
|--------------|---------------------------------------|
| <i>dirty</i> | Reference to dirty flag (user or plc) |
| <i>data</i>  | Data value reference (return)         |

Definition at line 297 of file plcBase.cpp.

References plc::dtString, plc::dtWString, mydata, mytype, and plc::reset\_and\_read().

Here is the call graph for this function:



### 8.14.3.19 Read() [5/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_wstring_value * data,
    size_type max ) const [protected]
```

Read character array (pwchar)

#### Parameters

|              |                                       |
|--------------|---------------------------------------|
| <i>dirty</i> | Reference to dirty flag (user or plc) |
| <i>data</i>  | Destination buffer                    |
| <i>max</i>   | Maximum length                        |

Definition at line 328 of file plcBase.cpp.

References Read().

Here is the call graph for this function:



### 8.14.3.20 ReadBinary()

```
DataValue::size_type plc::DataValue::ReadBinary (
    atomic_bool & dirty,
    type_binary p,
    size_type len ) const [protected]
```

Read data as binary

#### Parameters

|              |                                       |
|--------------|---------------------------------------|
| <i>dirty</i> | Reference to dirty flag (user or plc) |
| <i>p</i>     | value pointer (destination buffer)    |
| <i>len</i>   | Length in bytes                       |

Definition at line 400 of file plcBase.cpp.

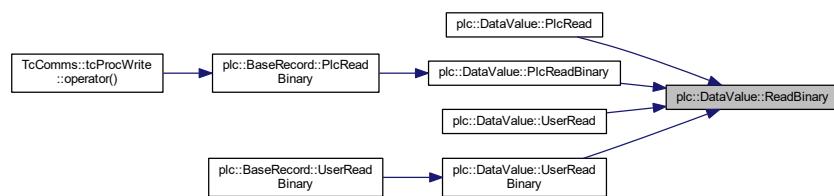
References plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, plc::DataValue<TypeDef::memory\_order, mydata, mysize, mytype, and Read().

Referenced by PlcRead(), PlcReadBinary(), UserRead(), and UserReadBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.14.3.21 SetValid()

```
void plc::DataValue::SetValid (
    atomic_bool & dirty,
    bool valid ) [protected]
```

Set the valid flag and set the dirty flag when flag changes

#### Parameters

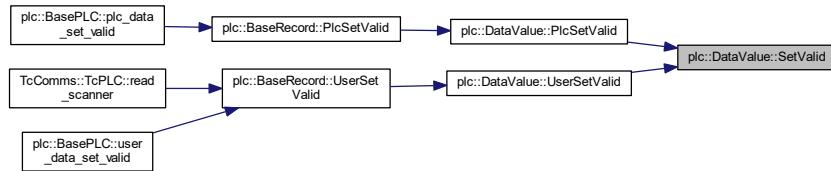
|              |  |
|--------------|--|
| <i>dirty</i> | Reference to dirty flag (user or plc)  |
| <i>valid</i> | True for valid data, False for invalid |

Definition at line 496 of file plcBase.cpp.

References plc::DataValueTypeDef::memory\_order, and myvalid.

Referenced by PlcSetValid(), and UserSetValid().

Here is the caller graph for this function:



#### 8.14.3.22 UserGetValid()

```
bool plc::DataValue::UserGetValid ( ) const [inline]
```

Get the valid flag and reset the dirty flag

##### Returns

valid True for valid data, False for invalid

Definition at line 316 of file plcBase.h.

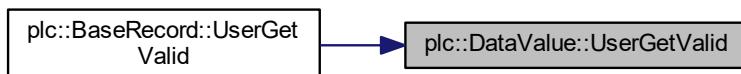
References GetValid(), and myplcdirty.

Referenced by plc::BaseRecord::UserGetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.14.3.23 UserRead() [1/4]

```
template<typename T >
bool plc::DataValue::UserRead (
    T & data ) const [inline]
```

Read data by the user

#### Parameters

|             |                               |
|-------------|-------------------------------|
| <i>data</i> | Data value reference (return) |
|-------------|-------------------------------|

Definition at line 260 of file plcBase.h.

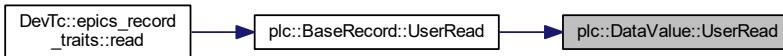
References myuserdirty, and Read().

Referenced by plc::BaseRecord::UserRead().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.14.3.24 UserRead() [2/4]

```
bool plc::DataValue::UserRead (
    type_string_value * data,
    size_type max ) const [inline]
```

Read character array (pchar) by the user

#### Parameters

|             |                    |
|-------------|--------------------|
| <i>data</i> | Destination buffer |
| <i>max</i>  | Maximum length     |

Definition at line 269 of file plcBase.h.

References myuserdirty, and Read().

Here is the call graph for this function:



#### 8.14.3.25 UserRead() [3/4]

```
template<size_type N>
bool plc::DataValue::UserRead (
    type_string_value(&) data[N] ) const [inline]
```

Read fixed length character array data by the user

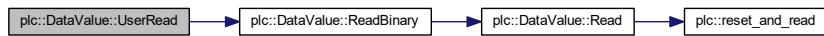
##### Parameters

|                   |  |
|-------------------|--|
| <code>data</code> | Data value reference for a fixed length character array (return) |
|-------------------|--|

Definition at line 264 of file plcBase.h.

References myuserdirty, and ReadBinary().

Here is the call graph for this function:



#### 8.14.3.26 UserRead() [4/4]

```
bool plc::DataValue::UserRead (
    type_wstring_value * data,
    size_type max ) const [inline]
```

Read character array (pwchar) by the user

## Parameters

|             |                    |
|-------------|--------------------|
| <i>data</i> | Destination buffer |
| <i>max</i>  | Maximum length     |

Definition at line 274 of file plcBase.h.

References myuserdirty, and Read().

Here is the call graph for this function:

**8.14.3.27 UserReadBinary()**

```
size_type plc::DataValue::UserReadBinary (
    type_binary p,
    size_type len ) const [inline]
```

Read data as binary by the user

## Parameters

|            |                                    |
|------------|------------------------------------|
| <i>p</i>   | value pointer (destination buffer) |
| <i>len</i> | Length in bytes                    |

Definition at line 299 of file plcBase.h.

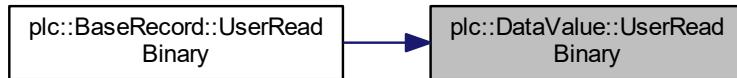
References myuserdirty, and ReadBinary().

Referenced by plc::BaseRecord::UserReadBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.14.3.28 UserSetValid()

```
void plc::DataValue::UserSetValid (
    bool valid ) [inline]
```

Set the valid flag and set the dirty flag when flag changes

##### Parameters

|              |  |
|--------------|--|
| <i>valid</i> | True for valid data, False for invalid |
|--------------|--|

Definition at line 313 of file plcBase.h.

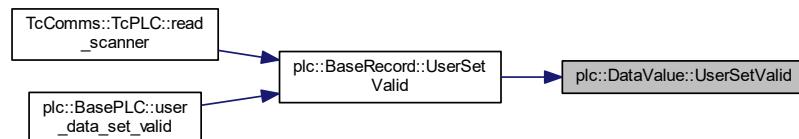
References myuserdirty, and SetValid().

Referenced by plc::BaseRecord::UserSetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.14.3.29 UserWrite() [1/4]

```
template<typename T >
bool plc::DataValue::UserWrite (
    const T & data ) [inline]
```

Write data by the user

#### Parameters

|             |                      |
|-------------|----------------------|
| <i>data</i> | Data value reference |
|-------------|----------------------|

Definition at line 278 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::UserWrite().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.14.3.30 UserWrite() [2/4]

```
bool plc::DataValue::UserWrite (
    const type_string_value * data,
    size_type max ) [inline]
```

Write character array (pchar) by the user

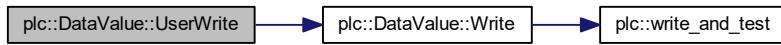
#### Parameters

|             |                |
|-------------|----------------|
| <i>data</i> | Source buffer  |
| <i>max</i>  | Maximum length |

Definition at line 288 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:



#### 8.14.3.31 UserWrite() [3/4]

```
template<size_type N>
bool plc::DataValue::UserWrite (
    const type_string_value(&) data[N] ) [inline]
```

Write fixed length character array data by the user

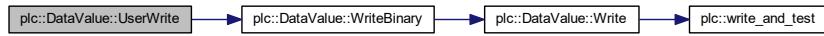
##### Parameters

|             |                      |
|-------------|----------------------|
| <i>data</i> | Data value reference |
|-------------|----------------------|

Definition at line 282 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Here is the call graph for this function:



#### 8.14.3.32 UserWrite() [4/4]

```
bool plc::DataValue::UserWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Write character array (wpchar) by the user

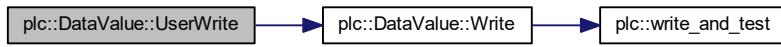
## Parameters

|             |                |
|-------------|----------------|
| <i>data</i> | Source buffer  |
| <i>max</i>  | Maximum length |

Definition at line 293 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:



### 8.14.3.33 UserWriteBinary()

```
size_type plc::DataValue::UserWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Write data as binary by the user

## Parameters

|            |                               |
|------------|-------------------------------|
| <i>p</i>   | value pointer (source buffer) |
| <i>len</i> | Length in bytes               |

Definition at line 304 of file plcBase.h.

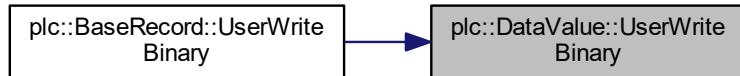
References myplcdirty, myuserdirty, and WriteBinary().

Referenced by plc::BaseRecord::UserWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.14.3.34 Write() [1/5]

```
template<typename T >
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const T & data ) [protected]
```

Write data

##### Parameters

|              |  |
|--------------|--|
| <i>dirty</i> | Reference to dirty flag (user or plc)        |
| <i>pend</i>  | Reference to pending read flag (plc or user) |
| <i>data</i>  | Data value reference (return)                |

[DataValue::UserWrite](#) (bool, Inegral and floating point types)

Definition at line 120 of file plcBaseTemplate.h.

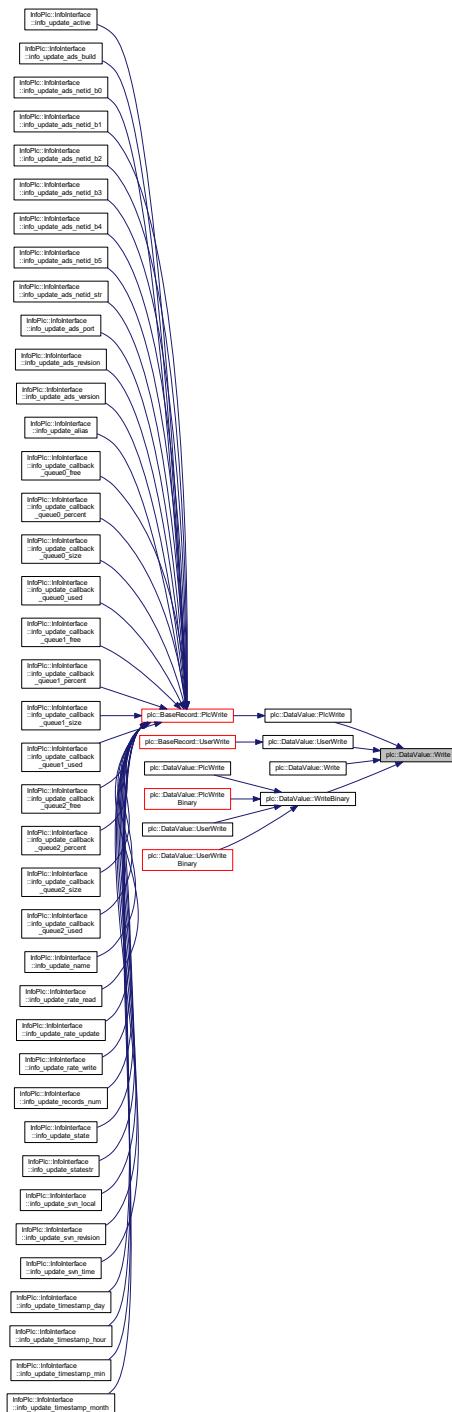
References `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `mydata`, `mytype`, `myvalid`, and `plc::write_and_test()`.

Referenced by `PlcWrite()`, `UserWrite()`, `Write()`, and `WriteBinary()`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.14.3.35 Write() [2/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
```

```
const atomic_bool & pend,
const type_string & data ) [protected]
```

Write string (template specialization)

#### Parameters

|              |  |
|--------------|--|
| <i>dirty</i> | Reference to dirty flag (user or plc)        |
| <i>pend</i>  | Reference to pending read flag (plc or user) |
| <i>data</i>  | Data value reference                         |

Definition at line 344 of file plcBase.cpp.

References plc::dtString, plc::dtWString, mydata, mytype, myvalid, and plc::write\_and\_test().

Here is the call graph for this function:



#### 8.14.3.36 Write() [3/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_string_value * data,
    size_type max ) [protected]
```

Write character array (pchar)

#### Parameters

|              |  |
|--------------|--|
| <i>dirty</i> | Reference to dirty flag (user or plc)        |
| <i>pend</i>  | Reference to pending read flag (plc or user) |
| <i>data</i>  | Source buffer                                |
| <i>max</i>   | Maximum length                               |

Definition at line 373 of file plcBase.cpp.

References Write().

Here is the call graph for this function:



#### 8.14.3.37 Write() [4/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_wstring & data ) [protected]
```

Write wstring (template specialization)

##### Parameters

|              |  |
|--------------|--|
| <i>dirty</i> | Reference to dirty flag (user or plc)        |
| <i>pend</i>  | Reference to pending read flag (plc or user) |
| <i>data</i>  | Data value reference                         |

Definition at line 360 of file plcBase.cpp.

References plc::dtWString, mydata, mytype, myvalid, and plc::write\_and\_test().

Here is the call graph for this function:



#### 8.14.3.38 Write() [5/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_wstring_value * data,
    size_type max ) [protected]
```

Write character array (pwchar)

**Parameters**

|              |  |
|--------------|--|
| <i>dirty</i> | Reference to dirty flag (user or plc)        |
| <i>pend</i>  | Reference to pending read flag (plc or user) |
| <i>data</i>  | Source buffer                                |
| <i>max</i>   | Maximum length                               |

Definition at line 386 of file plcBase.cpp.

References Write().

Here is the call graph for this function:

**8.14.3.39 WriteBinary()**

```
DataValue::size_type plc::DataValue::WriteBinary (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_binary p,
    size_type len ) [protected]
```

Write data as binary

**Parameters**

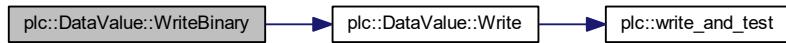
|              |  |
|--------------|--|
| <i>dirty</i> | Reference to dirty flag (user or plc)        |
| <i>pend</i>  | Reference to pending read flag (plc or user) |
| <i>p</i>     | value pointer (source buffer)                |
| <i>len</i>   | Length in bytes                              |

Definition at line 448 of file plcBase.cpp.

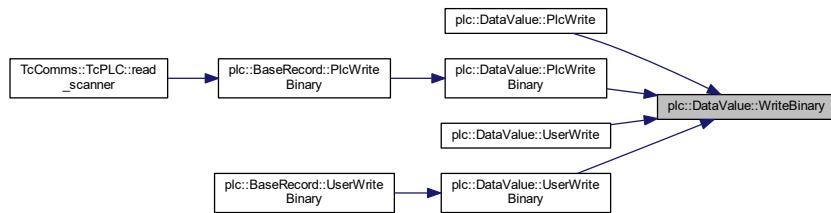
References plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, plc::DataValue<-TypeDef::memory\_order, mydata, mysize, mytype, and Write().

Referenced by PlcWrite(), PlcWriteBinary(), UserWrite(), and UserWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



## 8.14.4 Member Data Documentation

### 8.14.4.1 mysize

`size_type` `plc::DataValue::mysize` [protected]

Size of allocated memory for simple types; size of string class for strings and size of data for binary

Definition at line 465 of file `plcBase.h`.

Referenced by `get_size()`, `Init()`, `IsValid()`, `operator=()`, `ReadBinary()`, and `WriteBinary()`.

The documentation for this class was generated from the following files:

- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

## 8.15 `plc::DataValueTraits< T >` Struct Template Reference

Data value traits.

```
#include <plcBase.h>
```

## Public Types

- `typedef size_t size_type`  
*size type*
- `typedef plc::data_type_enum data_type_enum`  
*enumerated type for data type*
- `typedef T traits_type`  
*traits type*
- `typedef std::atomic< T > traits_atomic`  
*atomic variable type*

## Public Member Functions

- `const TValueTraits< TValueTypeDef::type_bool >::data_type_enum data_enum`  
*Bool specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_int8 >::data_type_enum data_enum`  
*Int8 specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_uint8 >::data_type_enum data_enum`  
*UInt8 specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_int16 >::data_type_enum data_enum`  
*Int16 specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_uint16 >::data_type_enum data_enum`  
*UInt16 specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_int32 >::data_type_enum data_enum`  
*Int32 specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_uint32 >::data_type_enum data_enum`  
*UInt32 specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_float >::data_type_enum data_enum`  
*Float specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_double >::data_type_enum data_enum`  
*Double specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_string >::data_type_enum data_enum`  
*String specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_wstring >::data_type_enum data_enum`  
*WString specialization for `TValueTraits`.*
- `const TValueTraits< TValueTypeDef::type_binary >::data_type_enum data_enum`  
*Binary specialization for `TValueTraits`.*

## Static Public Attributes

- `static const data_type_enum data_enum = dtInvalid`  
*data type enumertaion value*

### 8.15.1 Detailed Description

```
template<typename T>
struct plc::TValueTraits< T >
```

Data value traits.

Traits class for data value

Definition at line 112 of file plcBase.h.

## 8.15.2 Member Data Documentation

### 8.15.2.1 data\_enum

```
template<typename T >
const DataValueTraits< T >::data\_type\_enum plc::DataValueTraits< T >::data_enum = dtInvalid
[static]
```

data type enumertaion value

DataValue::data\_enum

Definition at line 123 of file plcBase.h.

The documentation for this struct was generated from the following files:

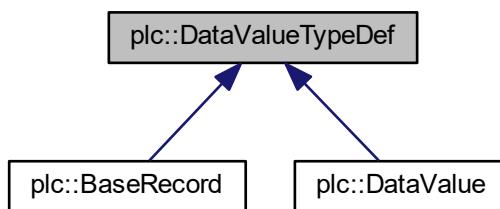
- [plcBase.h](#)
- [plcBaseTemplate.h](#)

## 8.16 plc::DataValueTypeDef Struct Reference

Collection of type definitions.

```
#include <plcBase.h>
```

Inheritance diagram for plc::DataValueTypeDef:



## Public Types

- `typedef size_t size_type`  
*size type*
- `typedef plc::data_type_enum data_type_enum`  
*enumerated type for data type*
- `typedef bool type_bool`  
*bool type*
- `typedef signed char type_int8`  
*1-byte integer type*
- `typedef unsigned char type_uint8`  
*1-byte unsigned integer type*
- `typedef short type_int16`  
*3-byte integer type*
- `typedef unsigned short type_uint16`  
*3-byte unsigned integer type*
- `typedef int type_int32`  
*4-byte integer type*
- `typedef unsigned int type_uint32`  
*4-byte unsigned integer type*
- `typedef long long type_int64`  
*8-byte integer type*
- `typedef unsigned long long type_uint64`  
*8-byte unsigned integer type*
- `typedef float type_float`  
*4-byte single precision floating point type*
- `typedef double type_double`  
*4-byte double precision floating point type*
- `typedef std::string type_string`  
*string type*
- `typedef std::wstring type_wstring`  
*wstring type*
- `typedef void * type_binary`  
*binary type*
- `typedef std::string::value_type type_string_value`  
*string character type*
- `typedef std::wstring::value_type type_wstring_value`  
*wstring character type*
- `typedef DataValueTraits< type_bool >::traits_atomic atomic_bool`  
*atomic bool type*
- `typedef DataValueTraits< type_int8 >::traits_atomic atomic_int8`  
*atomic 1-byte integer type*
- `typedef DataValueTraits< type_uint8 >::traits_atomic atomic_uint8`  
*atomic 1-byte unsigned integer type*
- `typedef DataValueTraits< type_int16 >::traits_atomic atomic_int16`  
*atomic 2-byte integer type*
- `typedef DataValueTraits< type_uint16 >::traits_atomic atomic_uint16`  
*atomic 2-byte unsigned integer type*
- `typedef DataValueTraits< type_int32 >::traits_atomic atomic_int32`  
*atomic 4-byte integer type*
- `typedef DataValueTraits< type_uint32 >::traits_atomic atomic_uint32`

- `atomic 4-byte unsigned integer type`
- `typedef DataValueTraits< type_int64 >::traits_atomic atomic_int64`  
`atomic 8-byte integer type`
- `typedef DataValueTraits< type_uint64 >::traits_atomic atomic_uint64`  
`atomic 8-byte unsigned integer type`
- `typedef DataValueTraits< type_float >::traits_atomic atomic_float`  
`atomic 4-byte single precision floating point type`
- `typedef DataValueTraits< type_double >::traits_atomic atomic_double`  
`atomic 8-byte double precision floating point type`
- `typedef DataValueTraits< type_string >::traits_atomic atomic_string`  
`atomic string type`
- `typedef DataValueTraits< type_wstring >::traits_atomic atomic_wstring`  
`atomic wstring type`
- `typedef DataValueTraits< type_binary >::traits_atomic atomic_binary`  
`atomic binary type`
- `typedef DataValueTypeDef::type_uint64 time_type`  
`Define timestamp type.`

## Static Public Attributes

- `static const std::memory_order memory_order = std::memory_order_seq_cst`  
`memory order used for atomic access`

### 8.16.1 Detailed Description

Collection of type definitions.

Type definitions for data value

Definition at line 129 of file plcBase.h.

The documentation for this struct was generated from the following file:

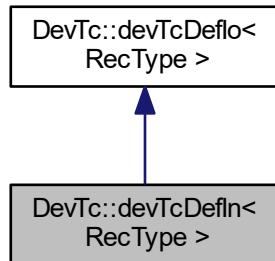
- `plcBase.h`

## 8.17 DevTc::devTcDefIn< RecType > Struct Template Reference

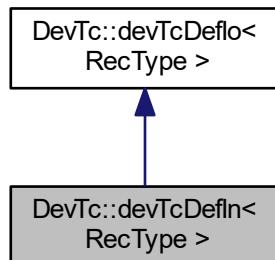
Device support input record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefIn< RecType >:



Collaboration diagram for DevTc::devTcDefIn< RecType >:



## Public Member Functions

- [devTcDefIn \(\)](#)  
*Constructor.*

## Static Public Member Functions

- static long [init\\_read\\_record \(rec\\_type\\_ptr prec\)](#)  
*init callback for read records*
- static long [read \(rec\\_type\\_ptr precord\)](#)  
*read callback*

## Additional Inherited Members

### 8.17.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefIn< RecType >
```

Device support input record.

Device Support Record for TwinCAT/ADS input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 361 of file devTc.h.

The documentation for this struct was generated from the following file:

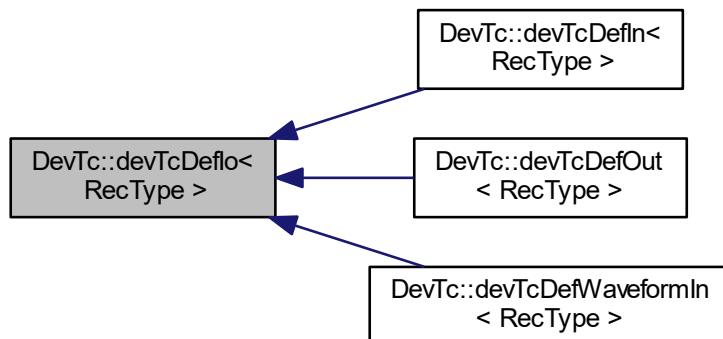
- [devTc.h](#)

## 8.18 DevTc::devTcDeflo< RecType > Struct Template Reference

Device support record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDeflo< RecType >:



## Public Types

- [typedef epics\\_record\\_traits< RecType >::traits\\_type rec\\_type](#)  
*Record type: aiRecord, etc.*
- [typedef rec\\_type \\* rec\\_type\\_ptr](#)  
*Pointer to record type.*

## Public Attributes

- long [number](#)  
*Number of support functions.*
- DEVSUPFUN [report\\_fn](#)  
*Report support function.*
- DEVSUPFUN [init\\_fn](#)  
*Init support function.*
- DEVSUPFUN [init\\_record\\_fn](#)  
*Record init support function.*
- DEVSUPFUN [get\\_ioint\\_info\\_fn](#)  
*IO/INT support function.*
- DEVSUPFUN [io\\_fn](#)  
*Read/write support function.*
- DEVSUPFUN [special\\_linconv\\_fn](#)  
*Linear conversion support function.*

## Protected Member Functions

- [devTcDeflo \(\)](#)  
*Hide constructor.*

## Static Protected Member Functions

- static long [get\\_ioint\\_info](#) (int cmd, dbCommon \*prec, IOSCANPVT \*ppvt)  
*IO/INT info callback.*

### 8.18.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDeflo< RecType >
```

Device support record.

Deviced Support Record for generic TwinCAT/ADS IO This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 326 of file devTc.h.

The documentation for this struct was generated from the following file:

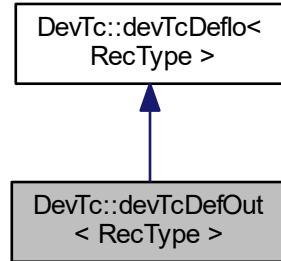
- [devTc.h](#)

## 8.19 DevTc::devTcDefOut< RecType > Struct Template Reference

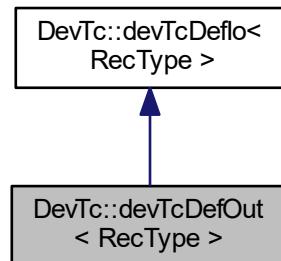
device support output record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefOut< RecType >:



Collaboration diagram for DevTc::devTcDefOut< RecType >:



### Public Member Functions

- [devTcDefOut \(\)](#)

*Constructor.*

### Static Public Member Functions

- static long [init\\_write\\_record \(rec\\_type\\_ptr prec\)](#)  
*init callback for write records*
- static long [write \(rec\\_type\\_ptr precord\)](#)  
*write callback*

## Additional Inherited Members

### 8.19.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefOut< RecType >
```

device support output record.

Device Support Record for TwinCAT/ADS output This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 377 of file devTc.h.

The documentation for this struct was generated from the following file:

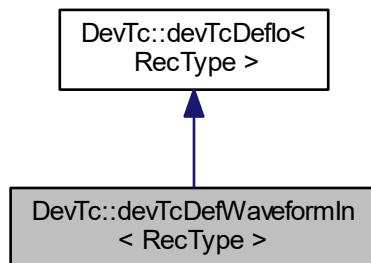
- [devTc.h](#)

## 8.20 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference

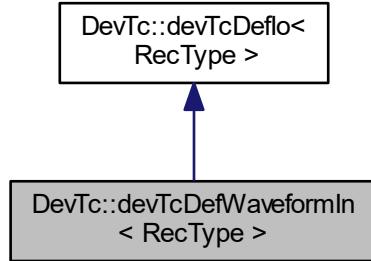
device support waveform record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefWaveformIn< RecType >:



Collaboration diagram for DevTc::devTcDefWaveformIn< RecType >:



## Public Member Functions

- [devTcDefWaveformIn \(\)](#)

*Constructor.*

## Static Public Member Functions

- static long [init\\_read\\_waveform\\_record \(rec\\_type\\_ptr prec\)](#)  
*init callback for read records*
- static long [read\\_waveform \(rec\\_type\\_ptr precord\)](#)  
*read callback*

## Additional Inherited Members

### 8.20.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefWaveformIn< RecType >
```

device support waveform record.

Device Support Record for TwinCAT/ADS waveform input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 393 of file devTc.h.

The documentation for this struct was generated from the following file:

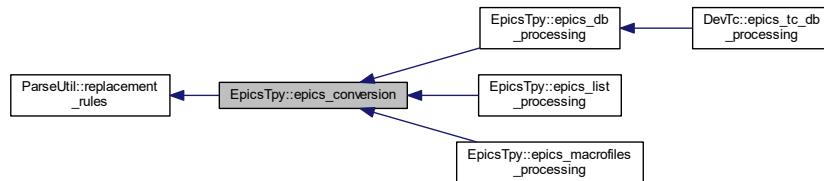
- [devTc.h](#)

## 8.21 EpicsTpy::epics\_conversion Class Reference

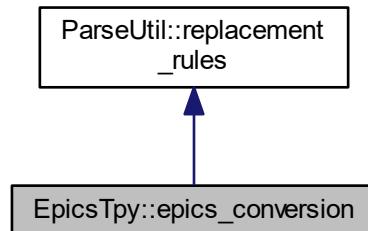
Epics conversion.

```
#include <TpToEpics.h>
```

Inheritance diagram for EpicsTpy::epics\_conversion:



Collaboration diagram for EpicsTpy::epics\_conversion:



### Public Member Functions

- `epics_conversion ()`  
*Default constructor.*
- `epics_conversion (case_type caseconv, bool noindex)`
- `epics_conversion (tc_epics_conv epics_conv, case_type caseconv, bool noldot, bool noindex)`
- `epics_conversion (int argc, const char *const argv[], bool argp[] = 0)`
- `int getopt (int argc, const char *const argv[], bool argp[] = 0)`
- `tc_epics_conv get_conversion_rule () const`  
*Get the conversion rule.*
- `void set_conversion_rule (tc_epics_conv epics_conv)`  
*Set the conversion rule.*
- `case_type get_case_rule () const`  
*Get the conversion rule.*
- `void set_case_rule (case_type epics_conv)`  
*Set the conversion rule.*

- bool [get\\_dot\\_rule \(\) const](#)  
*Get the leadin dot rule.*
- void [set\\_dot\\_rule \(bool noldot\)](#)  
*Set the leading dot rule.*
- bool [get\\_array\\_rule \(\) const](#)  
*Get the array index rule.*
- void [set\\_array\\_rule \(bool noindex\)](#)  
*Set the array conversion rule.*
- std::stringcase [get\\_prefix \(\) const](#)  
*Get the channel prefix.*
- void [set\\_prefix \(const std::stringcase &pvPrefix\)](#)  
*Set the channel prefix.*
- std::string [to\\_epics \(const std::stringcase &name\) const](#)

## Protected Attributes

- tc\_epics\_conv conv\_rule  
*Conversion rule.*
- case\_type case\_epics\_names  
*Case conversion rule.*
- bool no\_leading\_dot  
*Leading dot conversion rule.*
- bool no\_array\_index  
*Array index conversion rule.*
- std::stringcase prefix  
*Prefix to apply to all EPICS names.*

## Additional Inherited Members

### 8.21.1 Detailed Description

Epics conversion.

Epics channel conversion arguments Epics channels are generated from opc through a conversion rule

Definition at line 59 of file TpyToEpics.h.

### 8.21.2 Constructor & Destructor Documentation

#### 8.21.2.1 epics\_conversion() [1/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    case_type caseconv,
    bool noindex ) [inline]
```

Constructor

**Parameters**

|                 |   |
|-----------------|---|
| <i>caseconv</i> | Case conversion specification           |
| <i>noindex</i>  | Eliminate array indices '[n]' with '_n' |

Definition at line 68 of file TpyToEpics.h.

References EpicsTpy::ligo\_std.

**8.21.2.2 epics\_conversion() [2/3]**

```
EpicsTpy::epics_conversion::epics_conversion (
    tc_epics_conv epics_conv,
    case_type caseconv,
    bool noldot,
    bool noindex ) [inline]
```

Constructor

**Parameters**

|                   |   |
|-------------------|---|
| <i>epics_conv</i> | Epics conversion rule                   |
| <i>caseconv</i>   | Case conversion specification           |
| <i>noldot</i>     | Eliminate leading dot in a name         |
| <i>noindex</i>    | Eliminate array indices '[n]' with '_n' |

Definition at line 76 of file TpyToEpics.h.

**8.21.2.3 epics\_conversion() [3/3]**

```
EpicsTpy::epics_conversion::epics_conversion (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

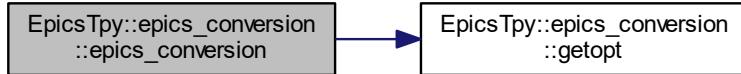
**Parameters**

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

Definition at line 87 of file TpyToEpics.h.

References getopt(), EpicsTpy::ligo\_std, and EpicsTpy::upper.

Here is the call graph for this function:



## 8.21.3 Member Function Documentation

### 8.21.3.1 getopt()

```
int EpicsTpy::epics_conversion::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/rn: Does not apply any special conversion rules /rd: Replaces dots with underscores in channel names /rl: LIGO standard conversion rule (default) /cp: Preserve case in EPICS channel names /cu: Force upper case in EPICS channel names (default) /cl: Force lower case in EPICS channel names /nd: Eliminates leading dot in channel name (default) /yd: Leaves leading dot in channel name /ni: Replaces array brackets with underscore (default) /yi: Leave array indices as is /p 'name': Include a prefix of 'name' for every channel (defaults to no prefix)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

#### Parameters

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

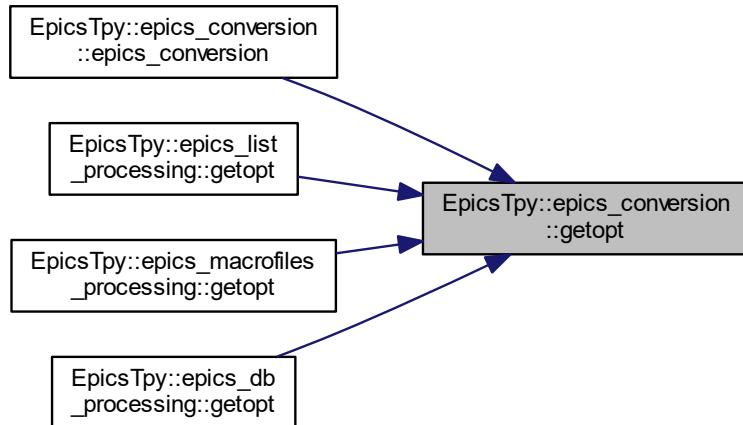
#### Returns

Number of arguments processed

Definition at line 24 of file TpyToEpics.cpp.

Referenced by `epics_conversion()`, `EpicsTpy::epics_list_processing::getopt()`, `EpicsTpy::epics_macrofiles_processing::getopt()`, and `EpicsTpy::epics_db_processing::getopt()`.

Here is the caller graph for this function:



### 8.21.3.2 `to_epics()`

```
string EpicsTpy::epics_conversion::to_epics (
    const std::string& name ) const
```

Converts a TwinCAT or OPC name to an EPICS channel name

#### Parameters

|                   |                  |
|-------------------|------------------|
| <code>name</code> | TwinCAT/opc name |
|-------------------|------------------|

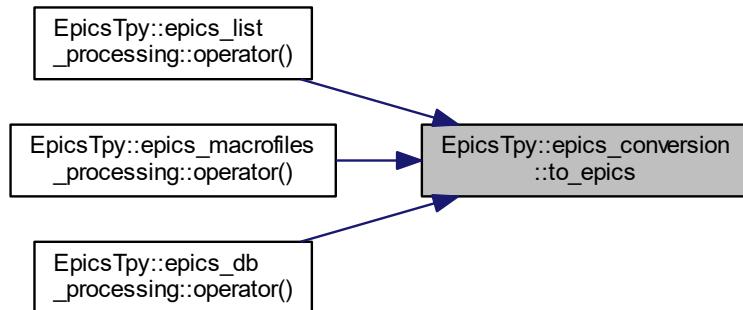
#### Returns

EPICS name

Definition at line 107 of file TpyToEpics.cpp.

Referenced by `EpicsTpy::epics_list_processing::operator()()`, `EpicsTpy::epics_macrofiles_processing::operator()()`, and `EpicsTpy::epics_db_processing::operator()()`.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

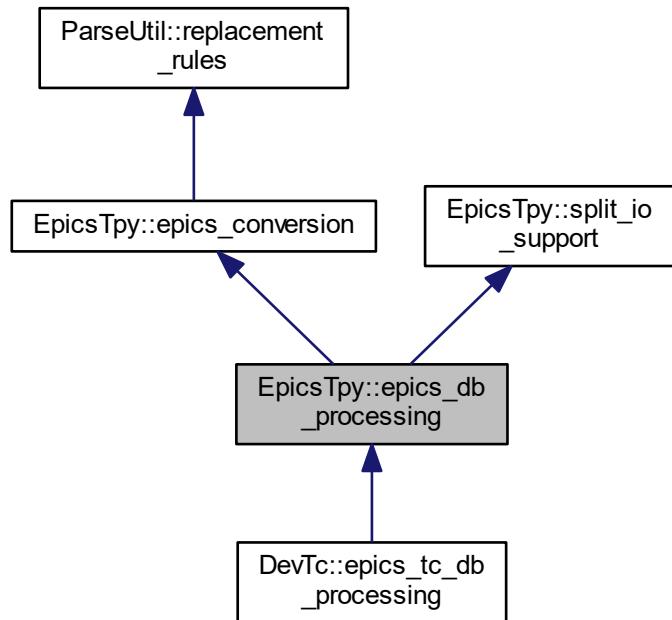
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

## 8.22 EpicsTpy::epics\_db\_processing Class Reference

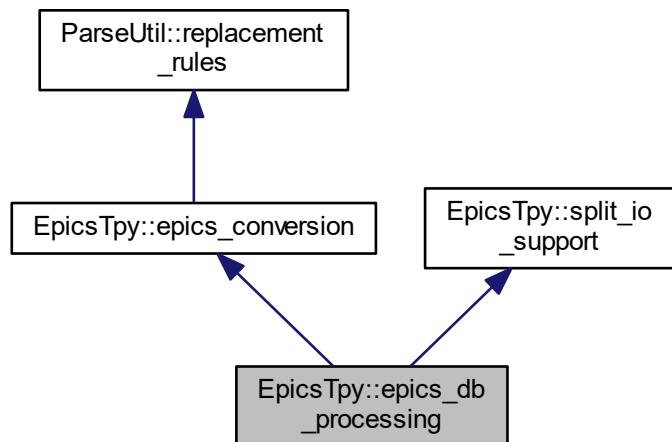
Epics database record processing

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics\_db\_processing:



Collaboration diagram for EpicsTpy::epics\_db\_processing:



## Public Member Functions

- [epics\\_db\\_processing \(\)](#)

*Default constructor.*

- [epics\\_db\\_processing](#) (const std::stringcase &fname, int argc, const char \*const argv[], bool argp[] = 0)
  - [int getopt](#) (int argc, const char \*const argv[], bool argp[] = 0)
  - [int my getopt](#) (int argc, const char \*const argv[], bool argp[] = 0)
  - [device\\_support\\_type get\\_device\\_support](#) () const
- Get device support conversion rule.*
- [void set\\_device\\_support](#) (device\_support\_type devsup)
- Set device support conversion rule.*
- [bool operator\(\)](#) (const ParseUtil::process\_arg &arg)

## Protected Member Functions

- [bool process\\_field\\_string](#) (std::stringcase name, std::stringcase val)
- [bool process\\_field\\_numeric](#) (std::stringcase name, int val)
- [bool process\\_field\\_numeric](#) (std::stringcase name, double val)
- [bool process\\_field\\_numeric](#) (std::stringcase name, std::stringcase val)
- [bool process\\_field\\_alarm](#) (std::stringcase name, std::stringcase severity)

## Protected Attributes

- [device\\_support\\_type device\\_support](#)
- Device support field conversion rule.*

## Additional Inherited Members

### 8.22.1 Detailed Description

pics database record processing

Class for generatig an EPICS database record

Definition at line 702 of file TpyToEpics.h.

### 8.22.2 Constructor & Destructor Documentation

#### 8.22.2.1 epics\_db\_processing()

```
EpicsTpy::epics_db_processing::epics_db_processing (
    const std::stringcase & fname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with [epics\\_conversion::getopt](#), [split\\_io\\_support::getopt](#) and [my getopt\(\)](#).

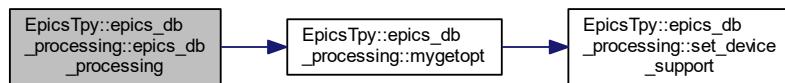
**Parameters**

|              |  |
|--------------|--|
| <i>fname</i> | Output filename  |
| <i>argc</i>  | Number of command line arguments   |
| <i>argv</i>  | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i>  | Excluded/processed arguments (in/out), array length must be argc         |

Definition at line 1212 of file TpyToEpics.cpp.

References [my getopt\(\)](#), and [EpicsTpy::tc\\_name](#).

Here is the call graph for this function:



## 8.22.3 Member Function Documentation

### 8.22.3.1 getopt()

```
int EpicsTpy::epics_db_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line Processed options with [epics\\_conversion::getopt](#) and my getopt.

**Parameters**

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

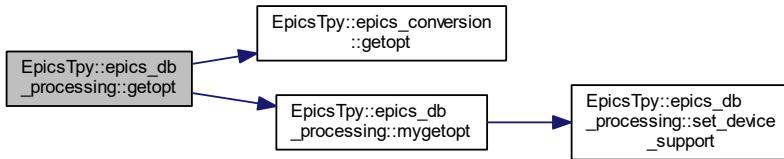
**Returns**

Number of arguments processed

Definition at line 1225 of file TpyToEpics.cpp.

References [EpicsTpy::epics\\_conversion::getopt\(\)](#), and [my getopt\(\)](#).

Here is the call graph for this function:



### 8.22.3.2 mygetopt()

```
int EpicsTpy::epics_db_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/devopc: Uses OPC name in INPUT/OUTPUT field (default) /devtc: Uses TwinCAT name in INPUT/OUTPUT fields instead of OPC

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

#### Parameters

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

#### Returns

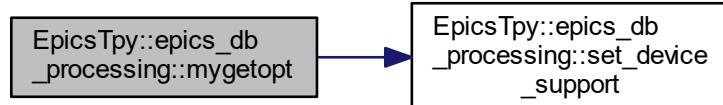
Number of arguments processed

Definition at line 1235 of file TpyToEpics.cpp.

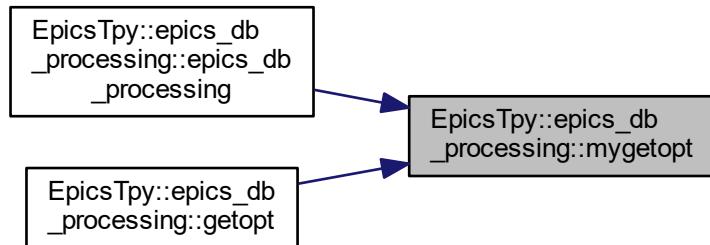
References `EpicsTpy::opc_name`, `set_device_support()`, and `EpicsTpy::tc_name`.

Referenced by `epics_db_processing()`, and `getopt()`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.22.3.3 operator()()

```
bool EpicsTpy::epics_db_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

#### Parameters

|                  |   |
|------------------|---|
| <code>arg</code> | Process argument describign the variable and type |
|------------------|---|

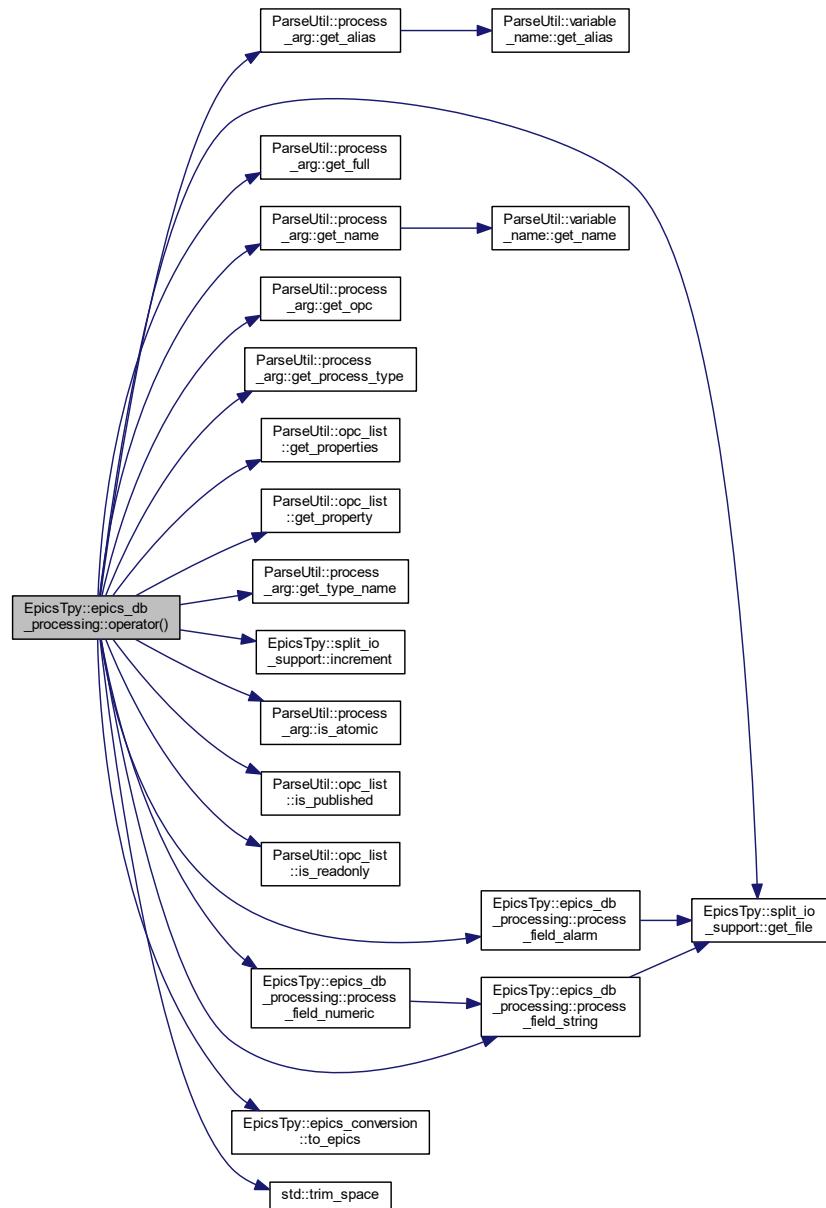
#### Returns

True if successfully processed

Definition at line 1268 of file TpyToEpics.cpp.

References device\_support, EpicsTpy::EPICS\_DB\_COSV, EpicsTpy::EPICS\_DB\_DESC, EpicsTpy::EPICS\_DB\_DRVH, EpicsTpy::EPICS\_DB\_DRVL, EpicsTpy::EPICS\_DB\_DTYP, EpicsTpy::EPICS\_DB\_EGU, EpicsTpy::EPICS\_DB\_HHSV, EpicsTpy::EPICS\_DB\_HIGH, EpicsTpy::EPICS\_DB\_HIHI, EpicsTpy::EPICS\_DB\_HOPR, EpicsTpy::EPICS\_DB\_HSV, EpicsTpy::EPICS\_DB\_HYST, EpicsTpy::EPICS\_DB\_INP, EpicsTpy::EPICS\_DB\_LLSV, EpicsTpy::EPICS\_DB\_LOLO, EpicsTpy::EPICS\_DB\_LOPR, EpicsTpy::EPICS\_DB\_LOW, EpicsTpy::EPICS\_DB\_LSV, EpicsTpy::EPICS\_DB\_MAJOR, EpicsTpy::EPICS\_DB\_MINOR, EpicsTpy::EPICS\_DB\_ONAM, EpicsTpy::EPICS\_DB\_OSV, EpicsTpy::EPICS\_DB\_OUT, EpicsTpy::EPICS\_DB\_PINI, EpicsTpy::EPICS\_DB\_PREC, EpicsTpy::EPICS\_DB\_SCAN, EpicsTpy::EPICS\_DB\_TSE, EpicsTpy::EPICS\_DB\_UNSV, EpicsTpy::EPICS\_DB\_ZNAM, EpicsTpy::EPICS\_DB\_ZRST, EpicsTpy::EPICS\_DB\_ZRSV, EpicsTpy::EPICS\_DB\_ZRVL, EpicsTpy::EPICS\_DB\_ZSV, ParseUtil::process\_arg::get\_alias(), EpicsTpy::split\_io\_support::get\_file(), ParseUtil::process\_arg::get\_full(), ParseUtil::process\_arg::get\_name(), ParseUtil::process\_arg::get\_opc(), ParseUtil::process\_arg::get\_process\_type(), ParseUtil::opc\_list::get\_properties(), ParseUtil::opc\_list::get\_property(), ParseUtil::process\_arg::get\_type\_name(), EpicsTpy::split\_io\_support::increment(), ParseUtil::process\_arg::is\_atomic(), ParseUtil::opc\_list::is\_published(), ParseUtil::opc\_list::is\_READONLY(), EpicsTpy::MAX\_EPICS\_CHANNEL, EpicsTpy::MAX\_EPICS\_DESC, EpicsTpy::opc\_name, ParseUtil::OPC\_PROP\_ALIAS, ParseUtil::OPC\_PROP\_ALMCOSV, ParseUtil::OPC\_PROP\_ALMDB, ParseUtil::OPC\_PROP\_ALMFFSV, ParseUtil::OPC\_PROP\_ALMH, ParseUtil::OPC\_PROP\_ALMHHSV, ParseUtil::OPC\_PROP\_ALMHHSV, ParseUtil::OPC\_PROP\_ALML, ParseUtil::OPC\_PROP\_ALMLL, ParseUtil::OPC\_PROP\_ALMLLSV, ParseUtil::OPC\_PROP\_ALMLSV, ParseUtil::OPC\_PROP\_ALMOSV, ParseUtil::OPC\_PROP\_ALMUNSV, ParseUtil::OPC\_PROP\_ALMZRSV, ParseUtil::OPC\_PROP\_ALMZSV, ParseUtil::OPC\_PROP\_CLOSE, ParseUtil::OPC\_PROP\_DESC, ParseUtil::OPC\_PROP\_DTYP, ParseUtil::OPC\_PROP\_FFST, ParseUtil::OPC\_PROP\_FIELD\_BEG, ParseUtil::OPC\_PROP\_FIELD\_END, ParseUtil::OPC\_PROP\_HIEU, ParseUtil::OPC\_PROP\_HIRANGE, ParseUtil::OPC\_PROP\_INOUT, ParseUtil::OPC\_PROP\_LOEU, ParseUtil::OPC\_PROP\_LORANGE, ParseUtil::OPC\_PROP\_OPEN, ParseUtil::OPC\_PROP\_PINI, ParseUtil::OPC\_PROP\_PLCNNAME, ParseUtil::OPC\_PROP\_PREC, ParseUtil::OPC\_PROP\_RECTYPE, ParseUtil::OPC\_PROP\_SERVER, ParseUtil::OPC\_PROP\_TSE, ParseUtil::OPC\_PROP\_UNIT, ParseUtil::OPC\_PROP\_ZRST, process\_field\_alarm(), process\_field\_numeric(), process\_field\_string(), EpicsTpy::tc\_name, EpicsTpy::epics\_conversion::to\_epics(), and std::trim\_space().

Here is the call graph for this function:



#### 8.22.3.4 process\_field\_alarm()

```

bool EpicsTpy::epics_db_processing::process_field_alarm (
    std::stringcase name,
    std::stringcase severity ) [protected]
  
```

Process a record field of type alarm

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <i>name</i>     | Name of field     |
| <i>severity</i> | Severity of alarm |

**Returns**

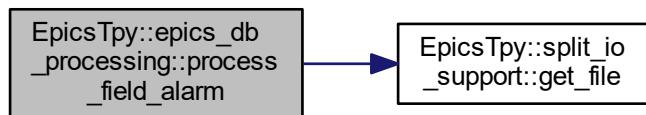
True if successful

Definition at line 1611 of file TpyToEpics.cpp.

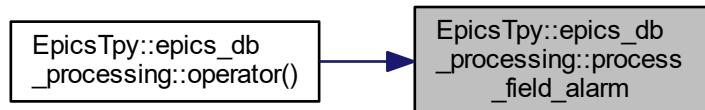
References `EpicsTpy::EPICS_DB_MAJOR`, `EpicsTpy::EPICS_DB_MINOR`, `EpicsTpy::EPICS_DB_NOALARM`, and `EpicsTpy::split_io_support::get_file()`.

Referenced by `operator()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**8.22.3.5 process\_field\_numeric() [1/3]**

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    double val )  [protected]
```

Process a record field of numeric type

**Parameters**

|             |                |
|-------------|----------------|
| <i>name</i> | Name of field  |
| <i>val</i>  | Value of field |

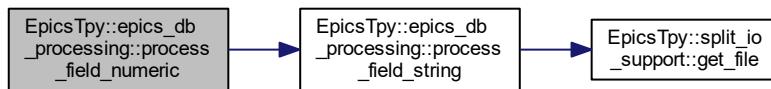
**Returns**

True if successful

Definition at line 1585 of file TpyToEpics.cpp.

References process\_field\_string().

Here is the call graph for this function:

**8.22.3.6 process\_field\_numeric() [2/3]**

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    int val )  [protected]
```

Process a record field of numeric type

**Parameters**

|             |                |
|-------------|----------------|
| <i>name</i> | Name of field  |
| <i>val</i>  | Value of field |

**Returns**

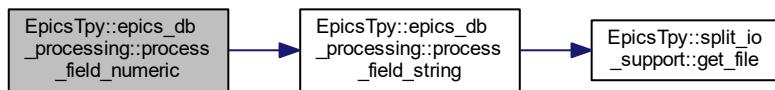
True if successful

Definition at line 1575 of file TpyToEpics.cpp.

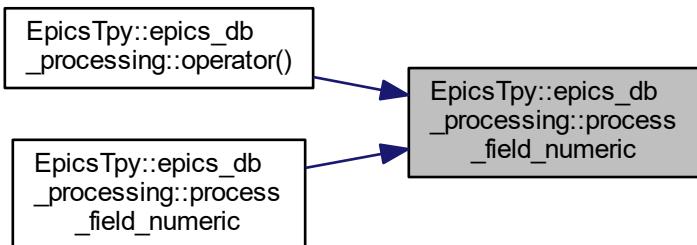
References process\_field\_string().

Referenced by operator()(), and process\_field\_numeric().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.22.3.7 process\_field\_numeric() [3/3]

```
bool EpicsTpyp::epics_db_processing::process_field_numeric (
    std::stringcase name,
    std::stringcase val )  [protected]
```

Process a record field of type string

#### Parameters

|             |                |
|-------------|----------------|
| <i>name</i> | Name of field  |
| <i>val</i>  | Value of field |

#### Returns

True if successful

Definition at line 1595 of file TpyToEpics.cpp.

References process\_field\_numeric().

Here is the call graph for this function:



### 8.22.3.8 process\_field\_string()

```
bool EpicsTpy::epics_db_processing::process_field_string (
    std::stringcase name,
    std::stringcase val ) [protected]
```

Process a record field of type string

#### Parameters

|             |                |
|-------------|----------------|
| <i>name</i> | Name of field  |
| <i>val</i>  | Value of field |

#### Returns

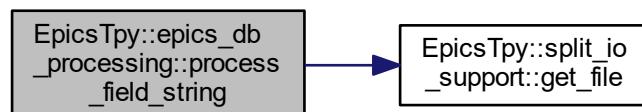
True if successful

Definition at line 1565 of file TpyToEpics.cpp.

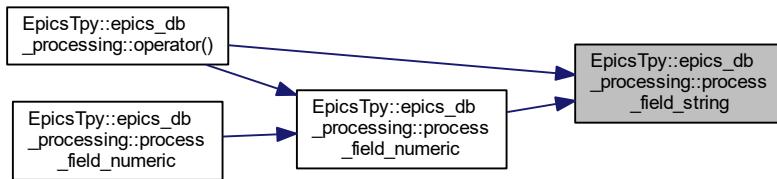
References [EpicsTpy::split\\_io\\_support::get\\_file\(\)](#).

Referenced by [operator\(\)\(\)](#), and [process\\_field\\_numeric\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

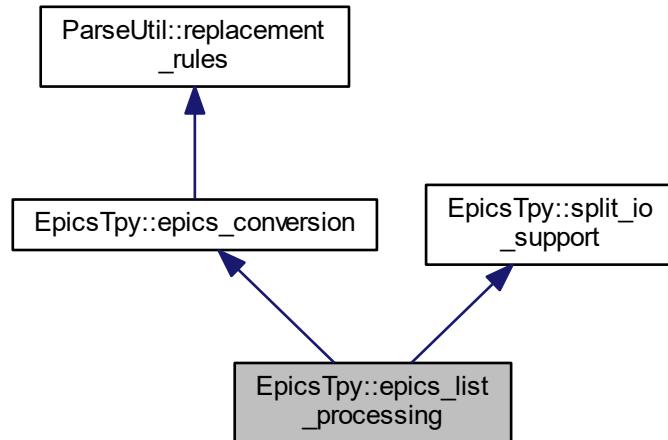
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

## 8.23 EpicsTpy::epics\_list\_processing Class Reference

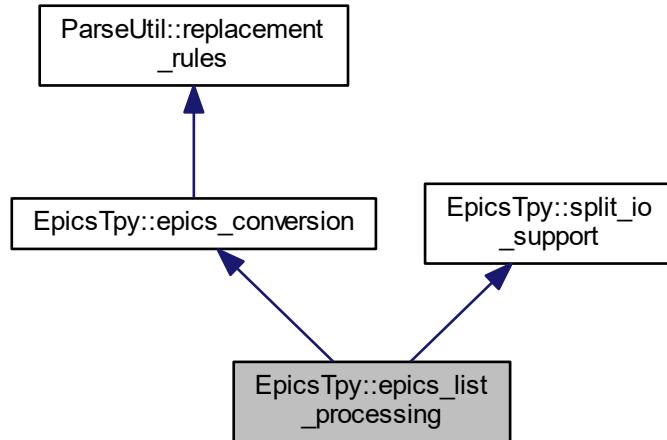
List processing.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics\_list\_processing:



Collaboration diagram for EpicsTpy::epics\_list\_processing:



## Public Member Functions

- [epics\\_list\\_processing \(\)](#)  
*Default constructor.*
- [epics\\_list\\_processing \(listing\\_type ltype, bool ll=false\)](#)
- [epics\\_list\\_processing \(const std::string& fname, int argc, const char \\*const argv\[\], bool argp\[\] = 0\)](#)
- [int getopt \(int argc, const char \\*const argv\[\], bool argp\[\] = 0\)](#)
- [int mygetopt \(int argc, const char \\*const argv\[\], bool argp\[\] = 0\)](#)
- [bool operator\(\) \(const ParseUtil::process\\_arg &arg\)](#)
- [listing\\_type get\\_listing \(\) const](#)  
*Get listing type.*
- [void set\\_listing \(listing\\_type lt\)](#)  
*Set listing.*
- [bool is\\_verbose \(\) const](#)  
*Is long listing?*
- [void set\\_verbose \(bool vrbs\)](#)  
*Set long listing.*

## Protected Attributes

- [listing\\_type listing](#)  
*Listing type.*
- [bool verbose](#)  
*long listing*

## Additional Inherited Members

### 8.23.1 Detailed Description

List processing.

Class for generating a channel list

Definition at line 435 of file TpyToEpics.h.

### 8.23.2 Constructor & Destructor Documentation

#### 8.23.2.1 epics\_list\_processing() [1/2]

```
EpicsTpy::epics_list_processing::epics_list_processing (
    listing_type ltype,
    bool ll = false )  [inline], [explicit]
```

Constructor

##### Parameters

|              |                 |
|--------------|-----------------|
| <i>ltype</i> | Type of listing |
| <i>ll</i>    | long listing    |

Definition at line 444 of file TpyToEpics.h.

#### 8.23.2.2 epics\_list\_processing() [2/2]

```
EpicsTpy::epics_list_processing::epics_list_processing (
    const std::string& fname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Constructor Command line arguments will override default parameters when specified. The format is the same as the arguments passed to the main program. argv[0] is program name and will be ignored. Processed options with [epics\\_conversion::getopt](#), [split\\_io\\_support::getopt](#) and [my getopt\(\)](#).

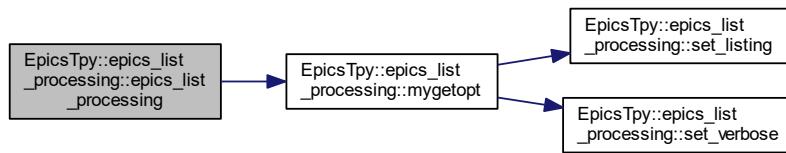
##### Parameters

|              |  |
|--------------|--|
| <i>fname</i> | Output filename  |
| <i>argc</i>  | Number of command line arguments   |
| <i>argv</i>  | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i>  | Excluded/processed arguments (in/out), array length must be argc         |

Definition at line 550 of file TpyToEpics.cpp.

References my getopt(), and EpicsTpy::standard.

Here is the call graph for this function:



### 8.23.3 Member Function Documentation

#### 8.23.3.1 getopt()

```
int EpicsTpy::epics_list_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line Processed options with [epics\\_conversion::getopt](#) and [my getopt\(\)](#).

##### Parameters

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

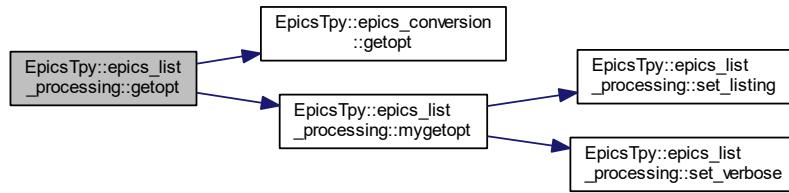
##### Returns

Number of arguments processed

Definition at line 563 of file TpyToEpics.cpp.

References EpicsTpy::epics\_conversion::getopt(), and [my getopt\(\)](#).

Here is the call graph for this function:



### 8.23.3.2 mygetopt()

```
int EpicsTpy::epics_list_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/I: Generates a standard listing (default) /Il: Generates a long listing /lb: Generates an autoburt save/restore file /li: Generates a LIGO DAQ ini file

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

#### Parameters

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

#### Returns

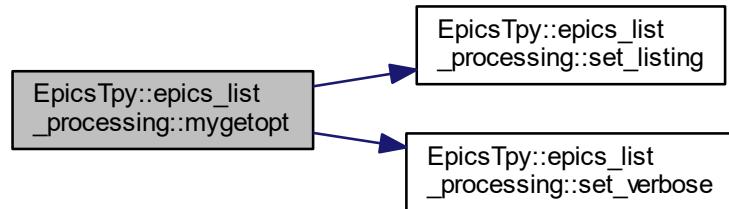
Number of arguments processed

Definition at line 573 of file TpyToEpics.cpp.

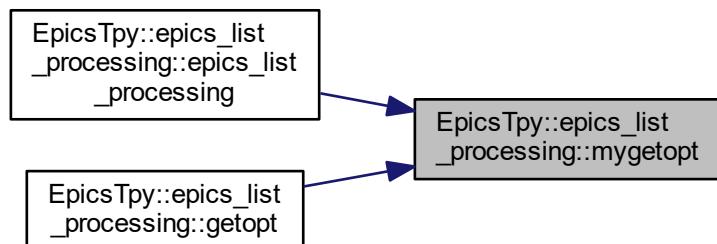
References EpicsTpy::autoburt, EpicsTpy::daqini, set\_listing(), set\_verbose(), and EpicsTpy::standard.

Referenced by epics\_list\_processing(), and getopt().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.23.3.3 operator()()

```
bool EpicsTpy::epics_list_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

#### Parameters

|                  |   |
|------------------|---|
| <code>arg</code> | Process argument describign the variable and type |
|------------------|---|

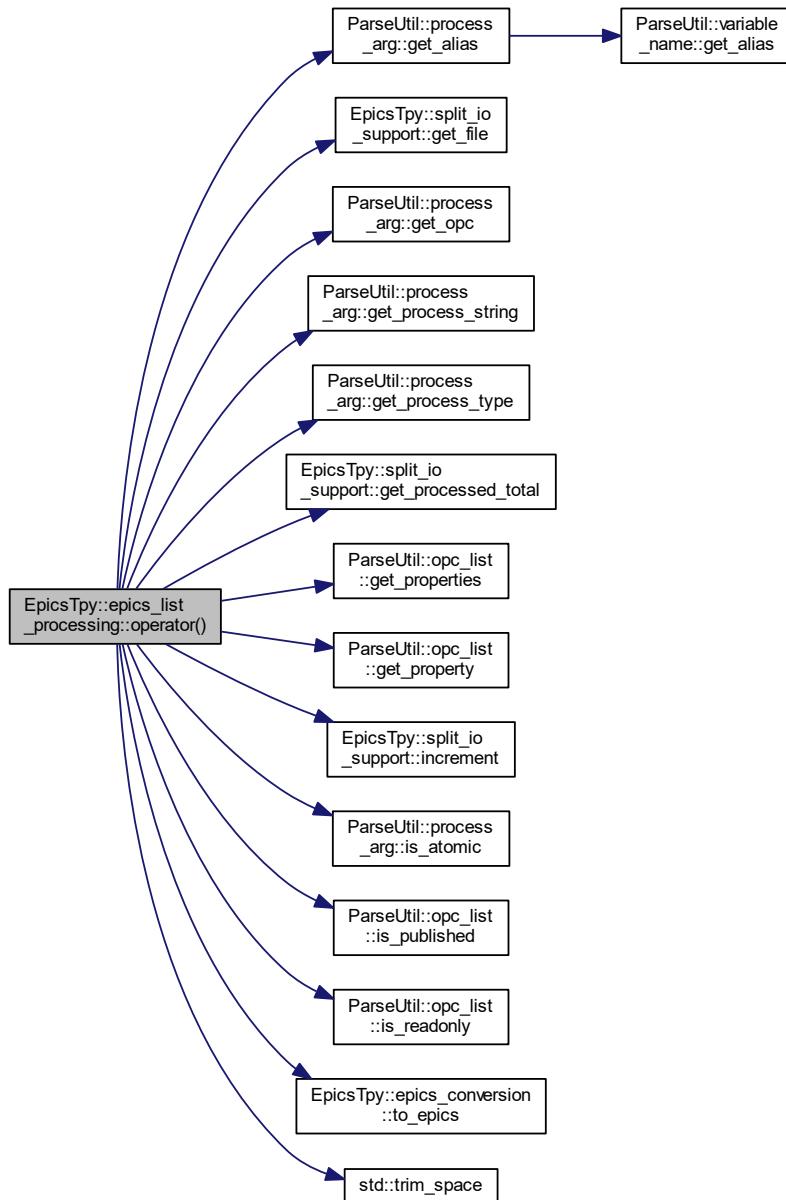
#### Returns

True if successfully processed

Definition at line 620 of file TpyToEpics.cpp.

References EpicsTpy::autoburt, EpicsTpy::daqini, ParseUtil::process\_arg::get\_alias(), EpicsTpy::split\_io\_support::get\_file(), ParseUtil::process\_arg::get\_opc(), ParseUtil::process\_arg::get\_process\_string(), ParseUtil::process\_arg::get\_process\_type(), EpicsTpy::split\_io\_support::get\_processed\_total(), ParseUtil::opc\_list::get\_properties(), ParseUtil::opc\_list::get\_property(), EpicsTpy::split\_io\_support::increment(), ParseUtil::process\_arg::is\_atomic(), ParseUtil::opc\_list::is\_published(), ParseUtil::opc\_list::is\_READONLY(), EpicsTpy::LIGODAQ\_DATATYPE\_DEFAULT, EpicsTpy::LIGODAQ\_DATATYPE\_FLOAT, EpicsTpy::LIGODAQ\_DATATYPE\_INT32, EpicsTpy::LIGODAQ\_DATYPE\_NAME, EpicsTpy::LIGODAQ\_INI\_HEADER, EpicsTpy::LIGODAQ\_UNIT\_DEFAULT, EpicsTpy::LIGODAQ\_UNIT\_NAME, EpicsTpy::LIGODAQ\_UNIT\_NONE, listing, ParseUtil::OPC\_PROP\_CLOSE, ParseUtil::OPC\_PROP\_FFST, ParseUtil::OPC\_PROP\_OPEN, ParseUtil::OPC\_PROP\_UNIT, ParseUtil::OPC\_PROP\_ZRST, EpicsTpy::standard, EpicsTpy::epics\_conversion::to\_epics(), std::trim\_space(), and verbose.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

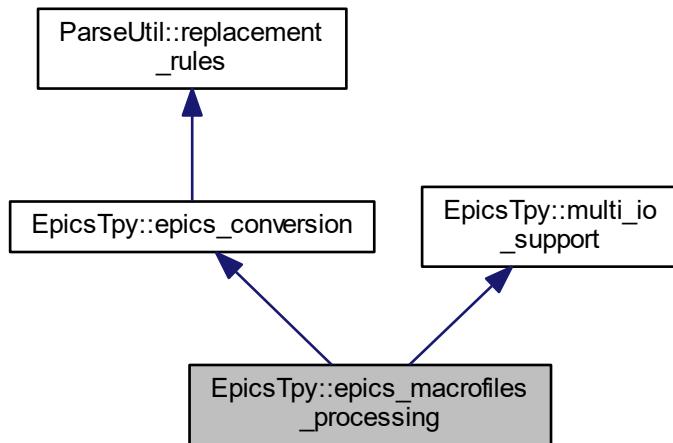
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

## 8.24 EpicsTpy::epics\_macrofiles\_processing Class Reference

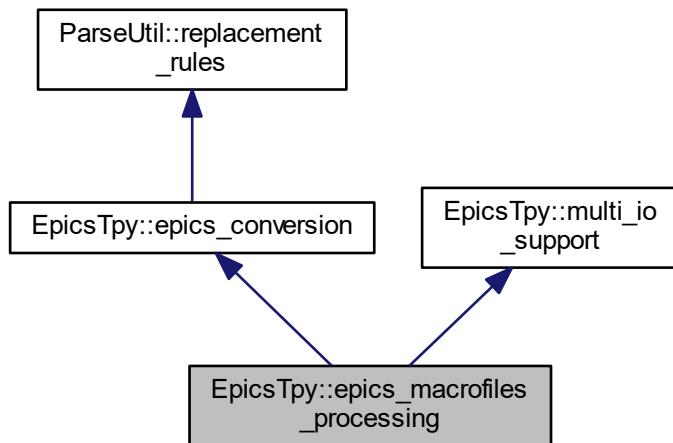
Macro file processing.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics\_macrofiles\_processing:



Collaboration diagram for EpicsTpy::epics\_macrofiles\_processing:



## Public Member Functions

- `epics_macrofiles_processing ()`  
*Default constructor.*
- `epics_macrofiles_processing (macrofile_type mt)`
- `epics_macrofiles_processing (const std::stringcase &pname, const std::stringcase &dbname, bool tcat3, int argc, const char *const argv[], bool argp[ ]=0)`
- `~epics_macrofiles_processing ()`  
*Destructor.*
- `void flush ()`  
*flush all pending processing*
- `int getopt (int argc, const char *const argv[], bool argp[ ]=0)`
- `int mygetopt (int argc, const char *const argv[], bool argp[ ]=0)`
- `bool operator() (const ParseUtil::process_arg &arg)`
- `macrofile_type get_macrofile_type () const`  
*Get listing type.*
- `void set_macrofile_type (macrofile_type m)`  
*Set listing.*
- `void set_plcname (const std::stringcase &name)`  
*Set PLC name.*
- `const std::stringcase & get_plcname () const`  
*Get PLC name.*
- `bool is_twincat3 () const`  
*Is this twincat 3?*
- `void set_twincat3 (bool tcat3=true)`  
*Set twincat 3 version?*
- `std::stringcase to_filename (const std::stringcase &epicsname)`  
*Translate epics name to filename.*
- `int get_processed_total () const`  
*Get number of processed channels.*

## Static Public Attributes

- static const `std::stringcase errorstruct = "ErrorStruct"`  
*Type name identifying an error struct ("ErrorStruct")*
- static const `std::stringcase errorlistext2 = "_Errors.exp"`  
*TwinCAT 2.11: File extension identifying a list of error msgs ("\_Errors.exp")*
- static const `std::regex errormatchregex2`  
*TwinCAT 2.11: Regular expression to match the entire error record definition.*
- static const `std::stringcase errorlistext31 = "_Errors.TcGVL"`  
*TwinCAT 3.1: File extension identifying a list of error msgs ("\_Errors.TcGVL")*
- static const `std::regex errormatchregex31`  
*TwinCAT 3.1: Regular expression to match the entire error record definition.*
- static const `std::regex errorsearchregex`  
*TwinCAT 2.11/3.1: Regular expression to search for the actual error messages.*

## Protected Member Functions

- `bool process_record (const macro_record &mrec, int level=0)`  
*Process top of stack.*

## Protected Attributes

- `macrofile_type macros`  
*Listing type.*
- `std::stringcase plcname`  
*PLC name.*
- `bool isTwinCAT3`  
*TwinCAT version.*
- `macro_stack procstack`  
*Processing stack.*
- `int rec_num`  
*Current number of processed channels (records)*
- `filename_set missing`  
*set of missing input files*

### 8.24.1 Detailed Description

Macro file processing.

Class for generating macro files to be used by medm

Definition at line 572 of file TpyToEpics.h.

### 8.24.2 Constructor & Destructor Documentation

#### 8.24.2.1 `epics_macrofiles_processing()` [1/2]

```
EpicsTpy::epics_macrofiles_processing::epics_macrofiles_processing (
    macrofile_type mt )  [inline], [explicit]
```

Constructor

Parameters

|                 |               |
|-----------------|---------------|
| <code>mt</code> | Type of macro |
|-----------------|---------------|

Definition at line 592 of file TpyToEpics.h.

#### 8.24.2.2 `epics_macrofiles_processing()` [2/2]

```
EpicsTpy::epics_macrofiles_processing::epics_macrofiles_processing (
    const std::stringcase & pname,
    const std::stringcase & dname,
```

```
    bool tcat3,
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with [epics\\_conversion::getopt](#), [multi\\_io\\_support::getopt](#) and [my getopt\(\)](#).

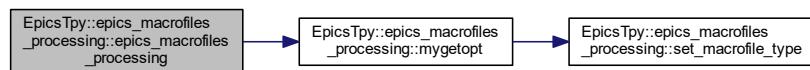
#### Parameters

|              |  |
|--------------|--|
| <i>pname</i> | PLC name   |
| <i>dname</i> | Directory name   |
| <i>tcat3</i> | True if we are processing TwinCAT 3.1 files                              |
| <i>argc</i>  | Number of command line arguments   |
| <i>argv</i>  | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i>  | Excluded/processed arguments (in/out), array length must be argc         |

Definition at line 724 of file TpyToEpics.cpp.

References ParseUtil::all, and my getopt().

Here is the call graph for this function:



### 8.24.3 Member Function Documentation

#### 8.24.3.1 getopt()

```
int EpicsTpy::epics_macrofiles_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )
```

Parse a command line Processed options with [epics\\_conversion::getopt](#) and [my getopt\(\)](#).

#### Parameters

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

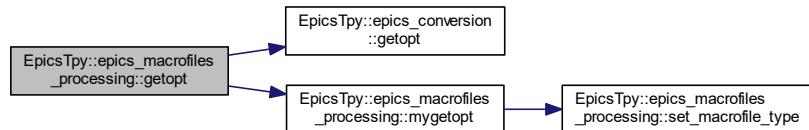
**Returns**

Number of arguments processed

Definition at line 749 of file TpyToEpics.cpp.

References EpicsTpy::epics\_conversion::getopt(), and my getopt().

Here is the call graph for this function:

**8.24.3.2 my getopt()**

```
int EpicsTpy::epics_macrofiles_processing::my getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/mf: Generate a macro file for each structure describing all fields /me: Generate a macro file for each structure describing the error messages /ma: Generate a macro file for each structure describing fields and errors (default)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

**Parameters**

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

**Returns**

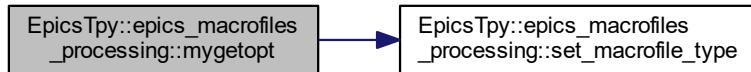
Number of arguments processed

Definition at line 759 of file TpyToEpics.cpp.

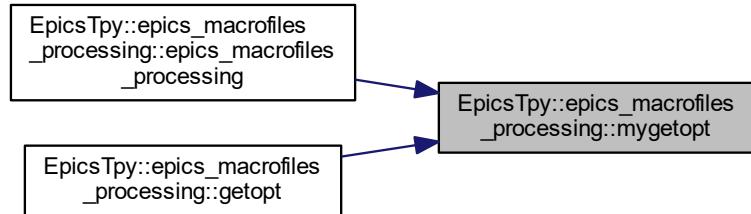
References EpicsTpy::all, EpicsTpy::errors, EpicsTpy::fields, and set\_macrofile\_type().

Referenced by epics\_macrofiles\_processing(), and getopt().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.24.3.3 operator()()

```
bool EpicsTpy::epics_macrofiles_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

#### Parameters

|            |   |
|------------|---|
| <i>arg</i> | Process argument describign the variable and type |
|------------|---|

#### Returns

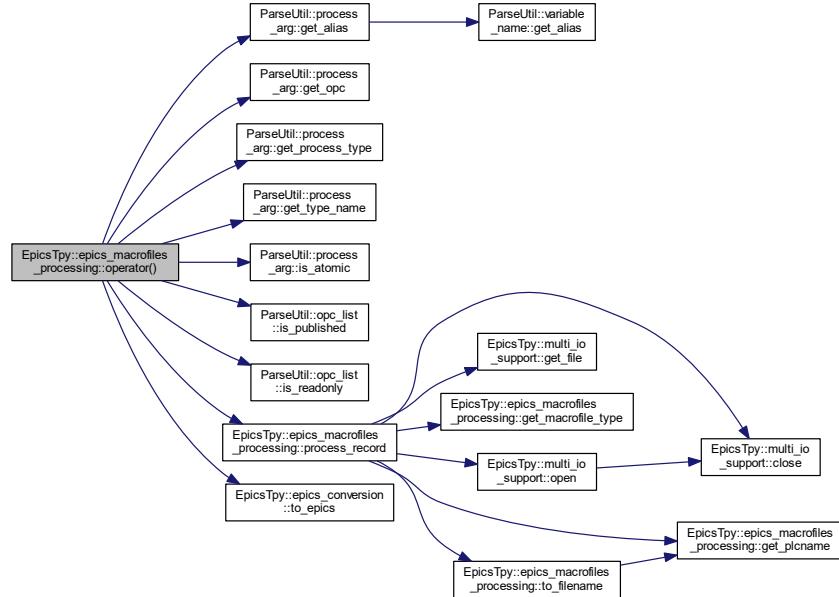
True if successfully processed

Definition at line 797 of file TpyToEpics.cpp.

References EpicsTpy::macro\_record::back, errorstruct, ParseUtil::process\_arg::get\_alias(), ParseUtil::process\_arg::get\_opc(), ParseUtil::process\_arg::get\_process\_type(), ParseUtil::process\_arg::get\_type\_name(), ParseUtil::process\_arg::is\_atomic(), ParseUtil::opc\_list::is\_published(), ParseUtil::opc\_list::is\_READONLY(), EpicsTpy::macro

`_record::iserror`, `EpicsTpy::macro_info::name`, `process_record()`, `procstack`, `EpicsTpy::macro_info::ptype`, `EpicsTpy::macro_info::readonly`, `rec_num`, `EpicsTpy::macro_record::record`, `EpicsTpy::epics_conversion::to_epics()`, and `EpicsTpy::macro_info::type_n`.

Here is the call graph for this function:



#### 8.24.3.4 to\_filename()

```
std::stringcase EpicsTpy::epics_macros_processing::to_filename (
    const std::stringcase & epicsname )
```

Translate epics name to filename.

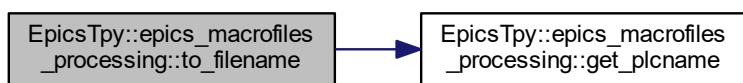
Translate epics name to filename

Definition at line 863 of file TpyToEpics.cpp.

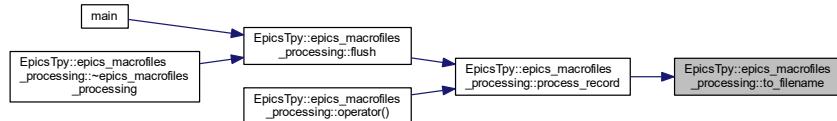
References `get_plcname()`, and `isTwinCAT3`.

Referenced by `process_record()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

## 8.25 DevTc::epics\_record\_traits< RecType > Struct Template Reference

Epics record traits.

```
#include <devTc.h>
```

### Classes

- struct [traits\\_type](#)

*Epics record type.*

### Public Types

- [typedef epicsFloat64 value\\_type](#)

*Value type of (raw) value field.*

### Static Public Member Functions

- static const char \*const [name \(\)](#)  
*Name of the record.*
- static [value\\_type \\* val \(traits\\_type \\*prec\)](#)  
*Returns the (raw) value of a record.*
- static bool [read \(traits\\_type \\*epicsrec, plc::BaseRecord \\*baserec\)](#)  
*Performs the read access on prec.*
- static bool [write \(plc::BaseRecord \\*baserec, traits\\_type \\*epicsrec\)](#)  
*Performs the write access on prec.*

## Static Public Attributes

- static const aitEnum `value_ait_type` = aitEnumFloat64
- static const aitInt32 `value_count` = 0
- static const int `value_conversion` = 0  
*return value for read\_io functions 0=default, 2=don't convert*
- static const bool `input_record` = true  
*Indicates if this is an input record.*
- static const bool `raw_record` = false  
*Indicates if this is a raw record.*

### 8.25.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::epics_record_traits< RecType >
```

Epics record traits.

This traits class for Epics records.

Definition at line 287 of file devTc.h.

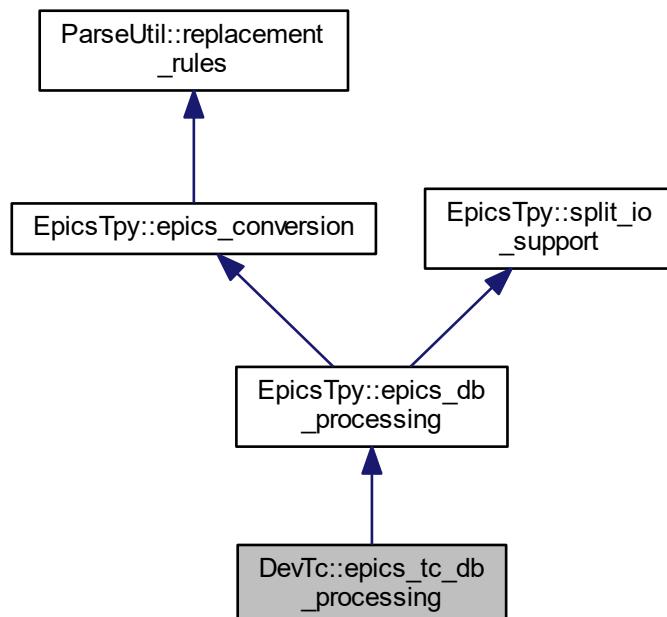
The documentation for this struct was generated from the following file:

- `devTc.h`

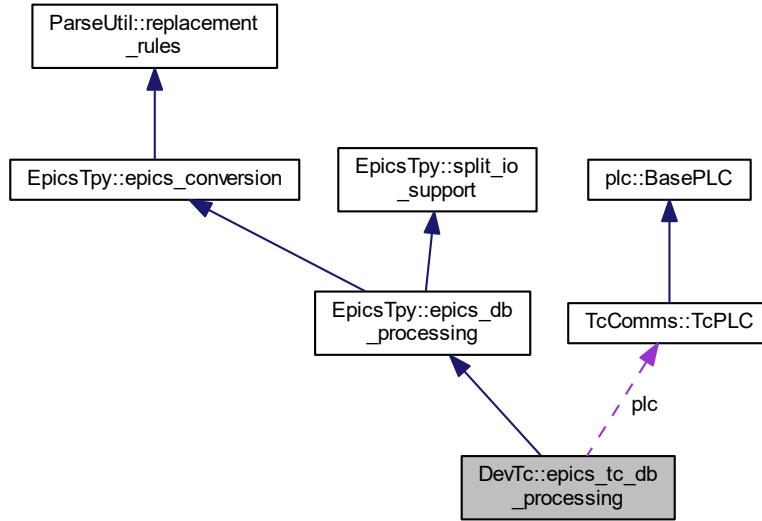
## 8.26 DevTc::epics\_tc\_db\_processing Class Reference

EPICS/TCat db processing.

Inheritance diagram for DevTc::epics\_tc\_db\_processing:



Collaboration diagram for DevTc::epics\_tc\_db\_processing:



## Public Member Functions

- `epics_tc_db_processing (TcComms::TcPLC &p, ParseUtil::replacement_table &rules, tc_listing_def *l=nullptr, tc_macro_def *m=nullptr)`
- `bool operator() (const ParseUtil::process_arg &arg)`
- `void flush ()`  
*Flush output files.*
- `int get_invalid_records () const`  
*Get number of EPICS records without tc records.*
- `bool patch_db_recordnames (std::stringcase &infodb)`

## Protected Member Functions

- `epics_tc_db_processing (const epics_tc_db_processing &)`  
*Disable copy constructor.*
- `epics_tc_db_processing & operator= (const epics_tc_db_processing &)`  
*Disable assignment operator.*
- `void init_lists ()`  
*Init lists.*
- `void done_lists ()`  
*Cleanup lists.*
- `bool process_lists (const ParseUtil::process_arg &arg)`
- `bool process_list (filename_rule_list_tuple &listdef, const ParseUtil::process_arg &arg)`
- `void init_macros ()`  
*Init macros.*
- `void done_macros ()`  
*Cleanup macros.*
- `bool process_macros (const ParseUtil::process_arg &arg)`
- `bool process_macro (dirname_arg_macro_tuple &macrodef, const ParseUtil::process_arg &arg)`

## Protected Attributes

- `TcComms::TcPLC * plc`  
*Pointer to PLC class.*
- `tc_listing_def * lists`  
*Pointer to a set of listings.*
- `tc_macro_def * macros`  
*Pointer to macros.*
- `int invnum`  
*Number of EPICS records without tc records.*

## Additional Inherited Members

### 8.26.1 Detailed Description

EPICS/TCat db processing.

Class for generating an EPICS database and tc record

Definition at line 105 of file drvTc.cpp.

### 8.26.2 Constructor & Destructor Documentation

#### 8.26.2.1 epics\_tc\_db\_processing()

```
DevTc::epics_tc_db_processing::epics_tc_db_processing (
    TcComms::TcPLC & p,
    ParseUtil::replacement_table & rules,
    tc_listing_def * l = nullptr,
    tc_macro_def * m = nullptr ) [inline], [explicit]
```

Default constructor

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>p</code>     | PLC                         |
| <code>rules</code> | Replacement rules           |
| <code>l</code>     | Pointer to list definitions |
| <code>m</code>     | Pointer to macro definition |

Definition at line 112 of file drvTc.cpp.

### 8.26.3 Member Function Documentation

### 8.26.3.1 operator()()

```
bool DevTc::epics_tc_db_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

#### Parameters

|            |   |
|------------|---|
| <i>arg</i> | Process argument describign the variable and type |
|------------|---|

#### Returns

True if successful

### 8.26.3.2 patch\_db\_recordnames()

```
bool DevTc::epics_tc_db_processing::patch_db_recordnames (
    std::stringcase & infodb )
```

Patch channel names in info database

#### Parameters

|               |                |
|---------------|----------------|
| <i>infodb</i> | EPICS database |
|---------------|----------------|

#### Returns

True if no errors

### 8.26.3.3 process\_list()

```
bool DevTc::epics_tc_db_processing::process_list (
    filename_rule_list_tuple & listdef,
    const ParseUtil::process_arg & arg ) [protected]
```

Process a listing

#### Parameters

|                |   |
|----------------|---|
| <i>listdef</i> | filename/rule pair defining a listing             |
| <i>arg</i>     | Process argument describign the variable and type |

**Returns**

True if successful

**8.26.3.4 process\_lists()**

```
bool DevTc::epics_tc_db_processing::process_lists (
    const ParseUtil::process_arg & arg ) [protected]
```

Process all listings

**Parameters**

|            |   |
|------------|---|
| <i>arg</i> | Process argument describign the variable and type |
|------------|---|

**Returns**

True if successful

**8.26.3.5 process\_macro()**

```
bool DevTc::epics_tc_db_processing::process_macro (
    dirname_arg_macro_tuple & macrodef,
    const ParseUtil::process_arg & arg ) [protected]
```

Process a macro

**Parameters**

|                 |   |
|-----------------|---|
| <i>macrodef</i> | filename/rule pair defining a macro               |
| <i>arg</i>      | Process argument describign the variable and type |

**Returns**

True if successful

**8.26.3.6 process\_macros()**

```
bool DevTc::epics_tc_db_processing::process_macros (
    const ParseUtil::process_arg & arg ) [protected]
```

Process all macros

**Parameters**

|            |   |
|------------|---|
| <i>arg</i> | Process argument describign the variable and type |
|------------|---|

**Returns**

True if successful

The documentation for this class was generated from the following file:

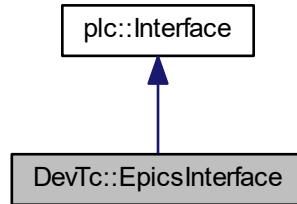
- [drvTc.cpp](#)

## 8.27 DevTc::EpicsInterface Class Reference

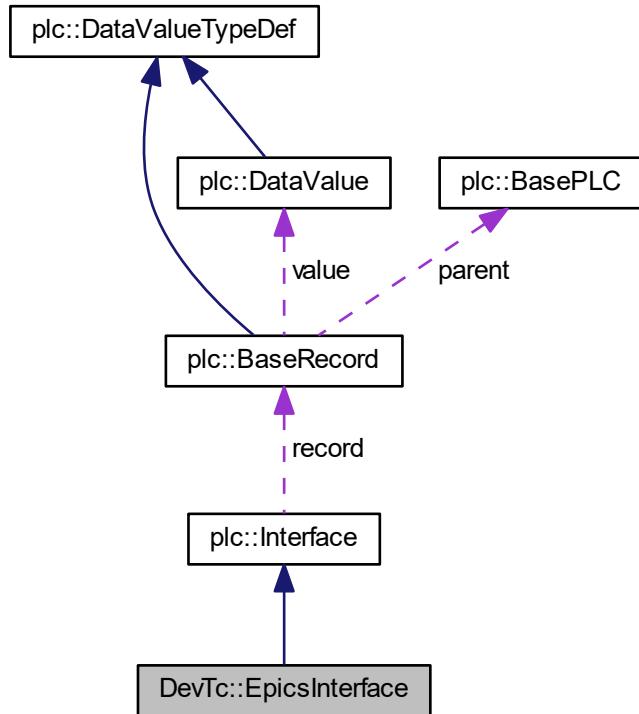
Epics interface class.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::EpicsInterface:



Collaboration diagram for DevTc::EpicsInterface:



## Public Member Functions

- `EpicsInterface (plc::BaseRecord &dval)`  
*Constructor.*
- `~EpicsInterface ()`  
*Deconstructor.*
- `void set_isPassive (bool passive)`  
*Set isPassive.*
- `bool get_isCallback () const`  
*Get isCallback.*
- `void set_isCallback (bool isCb)`  
*Set isCallback.*
- `void set_pEpicsRecord (dbCommon *pEpRecord)`  
*Set pEpicsRecord.*
- `bool get_callbackRequestPending () const`  
*Get callbackRequestPending.*
- `const CALLBACK & callback () const`  
*Get pointer to callback structure.*
- `CALLBACK & callback ()`  
*Get pointer to callback structure.*
- `const IOSCANPVT & ioscan () const`

- `IOSCANPVT & ioscan ()`

*Get reference to io scan list pointer.*
- `IOSCANPVT get_ioscan () const`

*Get reference to io scan list pointer.*
- `void set_ioscan (const IOSCANPVT ioscan)`

*Set pointer to io scan list.*
- `virtual bool push () override`

*Makes a call to the EPICS dbProcess function.*
- `virtual bool pull () override`

*Does nothing.*

## Static Public Member Functions

- `static int get_callback_queue_size (int pri)`
- `static int get_callback_queue_used (int pri)`
- `static int get_callback_queue_free (int pri)`

## Protected Member Functions

- `void ioscan_reset (int bitnum)`

*Reset ioscan use flag.*

## Protected Attributes

- `bool isPassive`
- `bool isCallback`
- `dbCommon * pEpicsRecord`

*Pointer to the EPICS record.*
- `std::mutex ioscanmux`

*IOSCAN mutex.*
- `IOSCANPVT ioscanpvt`

*Pointer to IO scan list.*
- `std::atomic< unsigned int > ioscan_inuse`

*Scan in progress (bit encoded value from priorities)*
- `CALLBACK callbackval`

*Callback structure.*

## Friends

- `void complete_io_scan (EpicsInterface *, IOSCANPVT, int)`

### 8.27.1 Detailed Description

Epics interface class.

This is a class for an EPICS Interface

Definition at line 94 of file devTc.h.

The documentation for this class was generated from the following files:

- [devTc.h](#)
- [devTc.cpp](#)

## 8.28 std::std::hash< std::stringcase > Struct Reference

hash for case insensitive string.

```
#include <stringcase_hash.h>
```

### Public Member Functions

- std::size\_t [operator\(\)](#) (const [stringcase](#) &str) const

### 8.28.1 Detailed Description

hash for case insensitive string.

This is a function specialization for case sensitive strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a string.

Definition at line 12 of file stringcase\_hash.h.

### 8.28.2 Member Function Documentation

#### 8.28.2.1 operator()()

```
std::size_t std::std::hash< std::stringcase >::operator() (
    const stringcase & str ) const
```

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from fnv\_32\_str and fnv\_64\_str <http://www.isthe.com/chongo/tech/comp/fnv/index.html#FNV-source>

#### Parameters

|                  |                |
|------------------|----------------|
| <code>str</code> | string to hash |
|------------------|----------------|

**Returns**

32/64 bit hash

The documentation for this struct was generated from the following file:

- [stringcase\\_hash.h](#)

## 8.29 std::std::hash< std::wstringcase > Struct Reference

hash for case insensitive unicode string.

```
#include <stringcase_hash.h>
```

### Public Member Functions

- `std::size_t operator()(const wstringcase &str) const`

#### 8.29.1 Detailed Description

hash for case insensitive unicode string.

This is a function specialization for case sensitive unicode strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a unicode string.

Definition at line 28 of file stringcase\_hash.h.

#### 8.29.2 Member Function Documentation

##### 8.29.2.1 operator()()

```
std::size_t std::std::hash< std::wstringcase >::operator() ( const wstringcase & str ) const
```

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from fnv\_32\_str <http://www.isthe.com/chongo/tech/comp/fnv/index.html#FNV-source>

**Parameters**

|                  |                |
|------------------|----------------|
| <code>str</code> | string to hash |
|------------------|----------------|

**Returns**

32/64 bit hash

The documentation for this struct was generated from the following file:

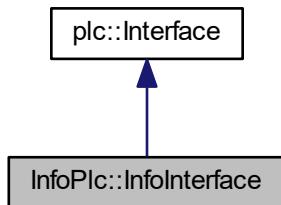
- [stringcase\\_hash.h](#)

## 8.30 InfoPlc::InfoInterface Class Reference

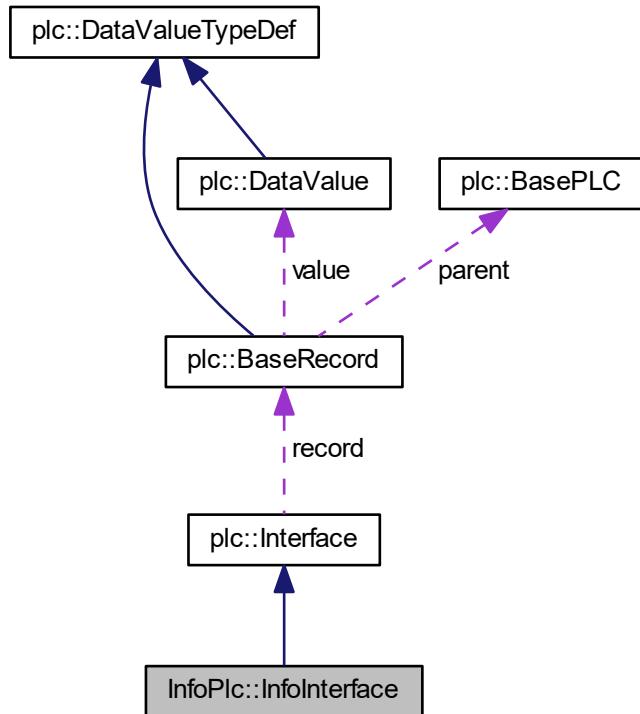
Info interface.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::InfoInterface:



Collaboration diagram for InfoPlc::InfoInterface:



## Public Member Functions

- `InfoInterface (plc::BaseRecord &dval)`
- `InfoInterface (plc::BaseRecord &dval, const std::stringcase &id, const std::stringcase &name, const std::stringcase &type)`
- `~InfoInterface ()`  
*Deconstructor.*
- `virtual bool push () override`  
*push data*
- `virtual bool pull () override`  
*pull data*
- `virtual bool update ()`  
*write data*
- `virtual void printVal (FILE *fp)`
- `virtual const char * get_symbol_name () const`

## Static Public Member Functions

- `template<class Function >`
- `static int get_infodb (const std::stringcase &prefix, const std::stringcase &plcaddr, Function &proc)`

## Protected Member Functions

- bool `info_update_name ()`  
*info update: Name of PLC*
- bool `info_update_alias ()`  
*info update: Alias name*
- bool `info_update_active ()`  
*info update: Running state of PLC*
- bool `info_update_state ()`  
*info update: AMS state of PLC*
- bool `info_update_statestr ()`  
*info update: AMS state of PLC*
- bool `info_update_timestamp_str ()`  
*info update: PLC time stamp*
- bool `info_update_timestamp_year ()`  
*info update: Year of PLC time stamp*
- bool `info_update_timestamp_month ()`  
*info update: Month of PLC time stamp*
- bool `info_update_timestamp_day ()`  
*info update: Day of PLC time stamp*
- bool `info_update_timestamp_hour ()`  
*info update: Hour of PLC time stamp*
- bool `info_update_timestamp_min ()`  
*info update: Minute of PLC time stamp*
- bool `info_update_timestamp_sec ()`  
*info update: Second of PLC time stamp*
- bool `info_update_rate_read ()`  
*info update: Period of read scanner in ms*
- bool `info_update_rate_write ()`  
*info update: Period of write scanner in ms*
- bool `info_update_rate_update ()`  
*info update: Period of update scanner in ms*
- bool `info_update_records_num ()`  
*info update: Number of EPICS records*
- bool `info_update_tpy_filename ()`  
*info update: Name of tpy file*
- bool `info_update_tpy_valid ()`  
*info update: Validity of tpy file*
- bool `info_update_tpy_time_str ()`  
*info update: Modification time of tpy file*
- bool `info_update_tpy_time_year ()`  
*info update: Year of tpy file time*
- bool `info_update_tpy_time_month ()`  
*info update: Month of tpy file time*
- bool `info_update_tpy_time_day ()`  
*info update: Day of tpy file time*
- bool `info_update_tpy_time_hour ()`  
*info update: Hour of tpy file time*
- bool `info_update_tpy_time_min ()`  
*info update: Minute of tpy file time*
- bool `info_update_tpy_time_sec ()`

- `bool info_update_ads_version ()`  
*info update: ADS library version*
- `bool info_update_ads_revision ()`  
*info update: ADS library revision*
- `bool info_update_ads_build ()`  
*info update: ADS library build*
- `bool info_update_ads_port ()`  
*info update: ADS/AMS port of PLC*
- `bool info_update_ads_netid_str ()`  
*info update: ADS/AMS address of PLC*
- `bool info_update_ads_netid_b0 ()`  
*info update: ADS/AMS address b0*
- `bool info_update_ads_netid_b1 ()`  
*info update: ADS/AMS address b1*
- `bool info_update_ads_netid_b2 ()`  
*info update: ADS/AMS address b2*
- `bool info_update_ads_netid_b3 ()`  
*info update: ADS/AMS address b3*
- `bool info_update_ads_netid_b4 ()`  
*info update: ADS/AMS address b4*
- `bool info_update_ads_netid_b5 ()`  
*info update: ADS/AMS address b5*
- `bool info_update_svn_local ()`  
*info update: SVN local modifications*
- `bool info_update_svn_revision ()`  
*info update: SVN revision, if fully committed*
- `bool info_update_svn_time ()`  
*info update: SVN compile time*
- `bool info_update_callback_queue0_size ()`  
*info update: Size of low priority callback queue*
- `bool info_update_callback_queue0_used ()`  
*info update: Used entries in low priority callback queue*
- `bool info_update_callback_queue0_free ()`  
*info update: Free entries low priority callback queue*
- `bool info_update_callback_queue0_percent ()`  
*info update: Usage percentage of low priority callback queue*
- `bool info_update_callback_queue1_size ()`  
*info update: Size of medium priority callback queue*
- `bool info_update_callback_queue1_used ()`  
*info update: Used entries in medium priority callback queue*
- `bool info_update_callback_queue1_free ()`  
*info update: Free entries medium priority callback queue*
- `bool info_update_callback_queue1_percent ()`  
*info update: Usage percentage of medium priority callback queue*
- `bool info_update_callback_queue2_size ()`  
*info update: Size of high priority callback queue*
- `bool info_update_callback_queue2_used ()`  
*info update: Used entries in high priority callback queue*
- `bool info_update_callback_queue2_free ()`  
*info update: Free entries high priority callback queue*
- `bool info_update_callback_queue2_percent ()`  
*info update: Usage percentage of high priority callback queue*

## Protected Attributes

- `std::stringcase tCatName`  
*Name of TCat symbol.*
- `std::stringcase tCatType`  
*Data type in TCat.*
- `update_enum update_freq`  
*Update frequency.*
- `info_update_method info_update`  
*pointer to info update method*

## Static Protected Attributes

- `static const info_dbrecord_list dbinfo_list`  
*List of db info records.*

### 8.30.1 Detailed Description

Info interface.

This is a class for a Info interface

Definition at line 55 of file infoPlc.h.

### 8.30.2 Constructor & Destructor Documentation

#### 8.30.2.1 InfoInterface() [1/2]

```
InfoPlc::InfoInterface::InfoInterface (
    plc::BaseRecord & dval ) [inline], [explicit]
```

Constructor

Parameters

|                   |   |
|-------------------|---|
| <code>dval</code> | BaseRecord that this interface is part of |
|-------------------|---|

Definition at line 60 of file infoPlc.h.

References InfoPlc::done.

### 8.30.2.2 InfoInterface() [2/2]

```
InfoPlc::InfoInterface::InfoInterface (
    plc::BaseRecord & dval,
    const std::stringcase & id,
    const std::stringcase & name,
    const std::stringcase & type )
```

Constructor

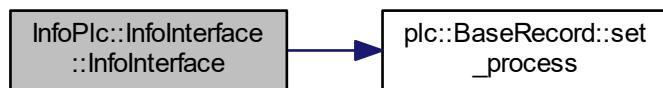
#### Parameters

|             |   |
|-------------|---|
| <i>dval</i> | BaseRecord that this interface is part of |
| <i>id</i>   | Short name info symbol                    |
| <i>name</i> | Full name of TCat symbol                  |
| <i>type</i> | Name of TCat data type                    |

Definition at line 523 of file infoPlc.cpp.

References dbinfo\_list, InfoPlc::done, info\_update, InfoPlc::once, plc::Interface::record, plc::BaseRecord::set\_process(), tCatType, and update\_freq.

Here is the call graph for this function:



## 8.30.3 Member Function Documentation

### 8.30.3.1 get\_infodb()

```
template<class Function >
int InfoPlc::InfoInterface::get_infodb (
    const std::stringcase & prefix,
    const std::stringcase & plcaddr,
    Function & proc ) [static]
```

Processes all info records

#### Parameters

|                |  |
|----------------|--|
| <i>prefix</i>  | Prefix to channel names in the info database |
| <i>plcaddr</i> | PLC associated with the info record          |
| <i>proc</i>    | database processing class                    |

**Returns**

Returns Number of info records processed

Definition at line 12 of file infoPlcTemplate.h.

References dbinfo\_list.

**8.30.3.2 get\_symbol\_name()**

```
virtual const char* InfoPlc::InfoInterface::get_symbol_name ( ) const [inline], [virtual]
```

Get symbol name

**Returns**

Symbol name associated with this interface

Reimplemented from [plc::Interface](#).

Definition at line 95 of file infoPlc.h.

References tCatName.

**8.30.3.3 printVal()**

```
void InfoPlc::InfoInterface::printVal (
    FILE * fp ) [virtual]
```

Prints TCat symbol value and information

**Parameters**

|           |                         |
|-----------|-------------------------|
| <i>fp</i> | File to print symbol to |
|-----------|-------------------------|

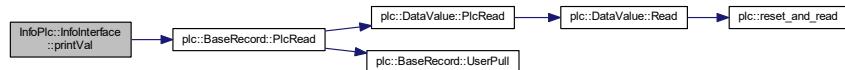
This is a function for printing the variable name and value of a record. Depending on the variable type, the readout from the ADS server is cast into the proper data type and printed to the output file fp.

Reimplemented from [plc::Interface](#).

Definition at line 572 of file infoPlc.cpp.

References [plc::BaseRecord::PlcRead\(\)](#), [plc::Interface::record](#), [tCatName](#), and [tCatType](#).

Here is the call graph for this function:



## 8.30.4 Member Data Documentation

### 8.30.4.1 dbinfo\_list

```
const info\_dbrecord\_list InfoPlc::InfoInterface::dbinfo_list [static], [protected]
```

List of db info records.

List of db info tuples.

Definition at line 212 of file `infoPlc.h`.

Referenced by `get_infodb()`, and `InfoInterface()`.

The documentation for this class was generated from the following files:

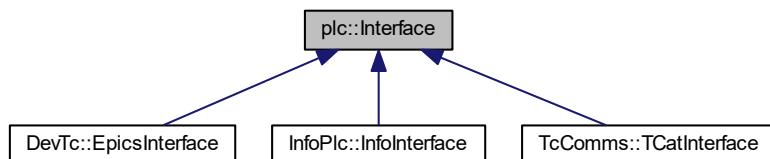
- [infoPlc.h](#)
- [infoPlc.cpp](#)
- [infoPlcTemplate.h](#)

## 8.31 plc::Interface Class Reference

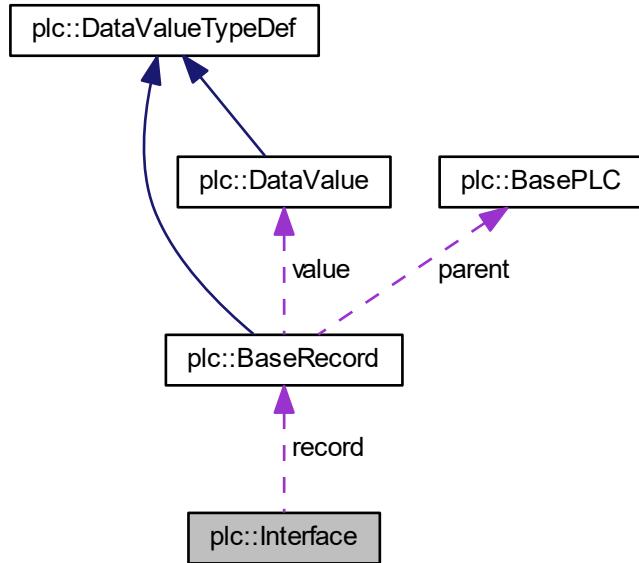
Abstract interface.

```
#include <plcBase.h>
```

Inheritance diagram for `plc::Interface`:



Collaboration diagram for `plc::Interface`:



## Public Member Functions

- `Interface (BaseRecord &dval)`
- `virtual ~Interface ()`  
*Destructor.*
- `BaseRecord & get_record ()`  
*Return a pointer to the tag/channel record.*
- `const BaseRecord & get_record () const`  
*Return a pointer to the tag/channel record.*
- `virtual bool push ()=0`  
*Pure virtual method indicating that the value needs to be pushed.*
- `virtual bool pull ()=0`  
*Pure virtual method indicating that the value needs to be pulled.*
- `BasePLC * get_parent ()`  
*Get parent PLC that owns this record.*
- `const BasePLC * get_parent () const`  
*Get parent PLC that owns this record.*
- `virtual void printVal (FILE *fp)`
- `virtual const char * get_symbol_name () const`  
*Get symbol name.*

## Protected Attributes

- `BaseRecord & record`  
*Pointer to tag/channel record associated with this interface.*

### 8.31.1 Detailed Description

Abstract interface.

This is a base class for an abstract interface to access the PLC (slave) or the user side (master).

Definition at line 33 of file plcBase.h.

### 8.31.2 Constructor & Destructor Documentation

#### 8.31.2.1 Interface()

```
plc::Interface::Interface (
    BaseRecord & dval ) [inline], [explicit]
```

Constructor

Parameters

|             |                                   |
|-------------|-----------------------------------|
| <i>dval</i> | Reference to a tag/channel record |
|-------------|-----------------------------------|

Definition at line 38 of file plcBase.h.

### 8.31.3 Member Function Documentation

#### 8.31.3.1 printVal()

```
virtual void plc::Interface::printVal (
    FILE * fp ) [inline], [virtual]
```

Print values to file

Parameters

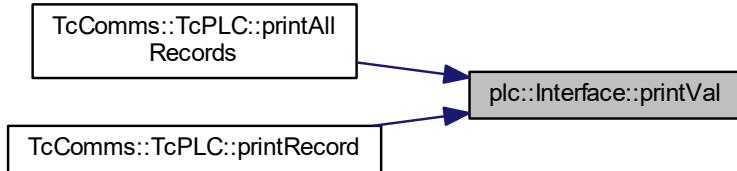
|           |              |
|-----------|--------------|
| <i>fp</i> | File pointer |
|-----------|--------------|

Reimplemented in [TcComms::TCatInterface](#), and [InfoPlc::InfoInterface](#).

Definition at line 58 of file plcBase.h.

Referenced by [TcComms::TcPLC::printAllRecords\(\)](#), and [TcComms::TcPLC::printRecord\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

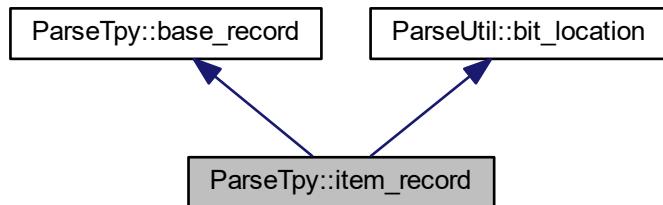
- [plcBase.h](#)
- [plcBase.cpp](#)

## 8.32 ParseTpy::item\_record Class Reference

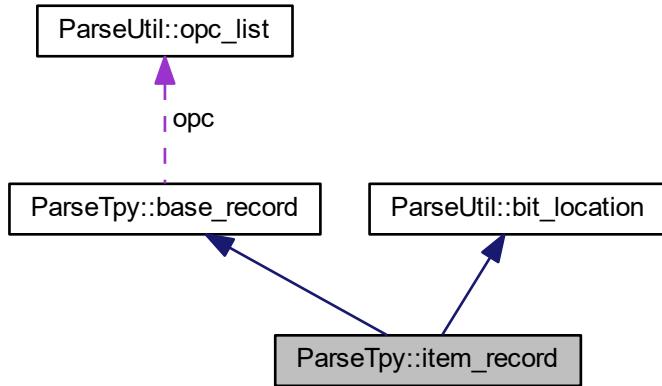
item record

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::item\_record:



Collaboration diagram for ParseTpy::item\_record:



## Public Member Functions

- [item\\_record \(\)](#)

*Default constructor.*

## Additional Inherited Members

### 8.32.1 Detailed Description

item record

This class stores typed items.

Definition at line 238 of file `ParseTpy.h`.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

## 8.33 EpicsTpy::macro\_info Struct Reference

Macro information.

```
#include <TpyToEpics.h>
```

## Public Member Functions

- [macro\\_info \(\)](#)  
*Default constructor.*

## Public Attributes

- [ParseUtil::process\\_type\\_enum ptype](#)  
*Process Type.*
- [std::stringcase name](#)  
*name of type*
- [std::stringcase type\\_n](#)  
*type definition*
- [bool readonly](#)  
*readonly*

### 8.33.1 Detailed Description

Macro information.

This structure describes a field

Definition at line 524 of file TpyToEpics.h.

The documentation for this struct was generated from the following file:

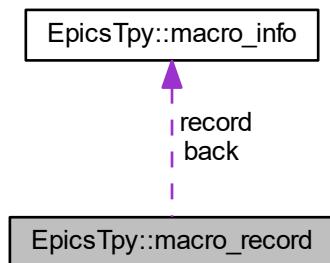
- [TpyToEpics.h](#)

## 8.34 EpicsTpy::macro\_record Struct Reference

Macro record.

```
#include <TpyToEpics.h>
```

Collaboration diagram for EpicsTpy::macro\_record:



## Public Attributes

- [macro\\_info record](#)  
*name of structure*
- [bool iserror](#)  
*is an ErrorStruct*
- [bool haserror](#)  
*contains an ErrorStruct*
- [int erroridx](#)  
*index if fields list to ErrorStruct*
- [macro\\_list fields](#)  
*List of fields.*
- [macro\\_info back](#)  
*name of upper level structure*

### 8.34.1 Detailed Description

Macro record.

This structure describes a record/struct

Definition at line 545 of file TpyToEpics.h.

The documentation for this struct was generated from the following file:

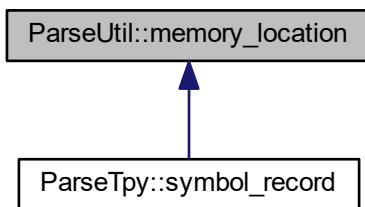
- [TpyToEpics.h](#)

## 8.35 ParseUtil::memory\_location Class Reference

Memory location.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::memory\_location:



## Public Member Functions

- `memory_location ()`  
*Default constructor.*
- `memory_location (int ig, int io, int bs)`
- `memory_location (const std::stringcase &s)`
- `bool isValid () const`  
*Validity.*
- `int get_igroup () const`  
*Get IGroup.*
- `void set_igroup (int ig)`  
*Set IGroup.*
- `int get_ioffset () const`  
*Get IOffset.*
- `void set_ioffset (int io)`  
*Set IOffset.*
- `int get_bytesize () const`  
*Get BitSize.*
- `void set_bytesize (int bs)`  
*Set BitSize.*
- `bool set_section (const bit_location &loc)`
- `std::stringcase get () const`
- `bool set (const std::stringcase &s)`

## Protected Attributes

- `int igrup`  
*Memory group.*
- `int ioffset`  
*Memory offset.*
- `int bytesize`  
*Memory size in bits.*

### 8.35.1 Detailed Description

Memory location.

This structure holds a memory location

Definition at line 263 of file ParseUtil.h.

### 8.35.2 Constructor & Destructor Documentation

#### 8.35.2.1 `memory_location()` [1/2]

```
ParseUtil::memory_location::memory_location (
    int ig,
    int io,
    int bs )  [inline]
```

Constructor

**Parameters**

|           |               |
|-----------|---------------|
| <i>ig</i> | Index group   |
| <i>io</i> | Index offset  |
| <i>bs</i> | Size in bytes |

Definition at line 272 of file ParseUtil.h.

**8.35.2.2 memory\_location() [2/2]**

```
ParseUtil::memory_location::memory_location (
    const std::string& s) [inline]
```

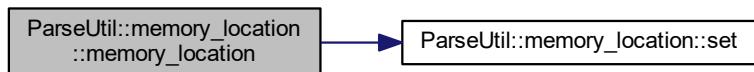
**Constructor****Parameters**

|          |                   |
|----------|-------------------|
| <i>s</i> | Definition string |
|----------|-------------------|

Definition at line 276 of file ParseUtil.h.

References set().

Here is the call graph for this function:

**8.35.3 Member Function Documentation****8.35.3.1 get()**

```
std::string ParseUtil::memory_location::get() const
```

Gets a string representation of a memory location

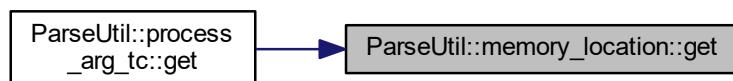
**Returns**

string with format "igroup/ioffset:size", empty on error

Definition at line 267 of file ParseUtil.cpp.

Referenced by ParseUtil::process\_arg\_tc::get().

Here is the caller graph for this function:

**8.35.3.2 set()**

```
bool ParseUtil::memory_location::set (
    const std::string& s )
```

Set the memory location using a string of the form: "igroup/ioffset:size" where the size is in bytes

**Parameters**

|   |                                   |
|---|-----------------------------------|
| s | String describing memory location |
|---|-----------------------------------|

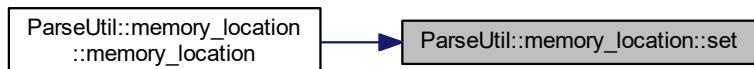
**Returns**

True if successful

Definition at line 280 of file ParseUtil.cpp.

Referenced by memory\_location().

Here is the caller graph for this function:



### 8.35.3.3 set\_section()

```
bool ParseUtil::memory_location::set_section (
    const bit_location & loc )
```

Set a sub section

#### Parameters

|            |                                   |
|------------|-----------------------------------|
| <i>loc</i> | Bit location within memory region |
|------------|-----------------------------------|

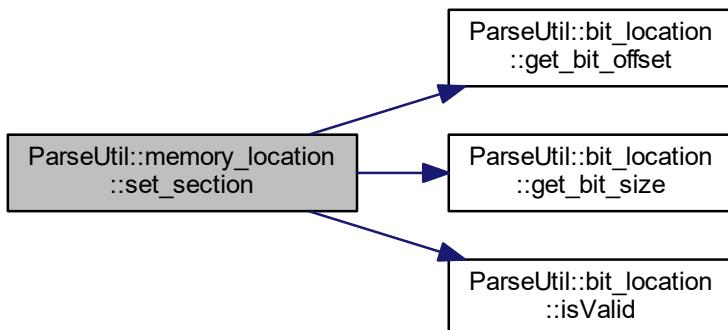
#### Returns

True if section within bounds, false otherwise

Definition at line 252 of file ParseUtil.cpp.

References ParseUtil::bit\_location::get\_bit\_offset(), ParseUtil::bit\_location::get\_bit\_size(), and ParseUtil::bit\_location::isValid().

Here is the call graph for this function:



The documentation for this class was generated from the following files:

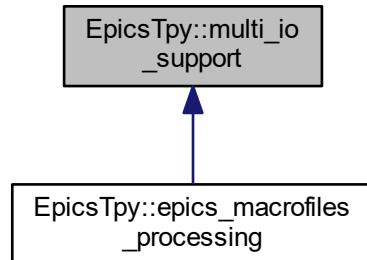
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

## 8.36 EpicsTpy::multi\_io\_support Class Reference

Multiple IO support.

```
#include <TpToEpics.h>
```

Inheritance diagram for EpicsTpy::multi\_io\_support:



## Public Member Functions

- `multi_io_support ()`  
*Default constructor.*
- `multi_io_support (const std::stringcase &dbname)`  
*Constructor.*
- `multi_io_support (const std::stringcase &dbname, int argc, const char *const argv[], bool argp[] = 0)`
- `~multi_io_support ()`  
*Destructor.*
- `bool operator! () const`  
*Return error.*
- `bool open (const std::stringcase &fname, const std::stringcase &io = "w", bool superrmsg = false)`  
*Open file for reading/writing.*
- `void close ()`  
*Close file.*
- `FILE * get_file () const`  
*Get file handle.*
- `void set_outdirname (const std::stringcase &dbname)`  
*Set output directory name.*
- `const std::stringcase & get_outdirname () const`  
*Get output directory name.*
- `void set_in dirname (const std::stringcase &dbname)`  
*Set input directory name.*
- `const std::stringcase & get_in dirname () const`  
*Get input directory name.*
- `const std::stringcase & get_filename () const`  
*Get full filename.*
- `io_filestat fileread () const`  
*Reading.*
- `int get_filein_total () const`  
*Get number of read files.*
- `int get_fileout_total () const`  
*Get number of written files.*

## Protected Member Functions

- void `set_filename` (const `std::stringcase` &fname)  
*Set output filename.*
- int `getopt` (int argc, const char \*const argv[], bool argp[] = 0)

## Protected Attributes

- `std::stringcase outdirname`  
*Directory name.*
- `std::stringcase indirname`  
*Directory name.*
- `std::stringcase filename`  
*Current filename.*
- `io_filestat filestat`  
*reading or writing?*
- FILE \* `filehandle`  
*Output file.*
- int `file_num_in`  
*Current file number of processed read only channels (records)*
- int `file_num_out`  
*Current file number of processed input/output channels (records)*

### 8.36.1 Detailed Description

Multiple IO support.

Multi file IO support Supports a directory argument and opens files within

Definition at line 321 of file TpyToEpics.h.

### 8.36.2 Constructor & Destructor Documentation

#### 8.36.2.1 multi\_io\_support()

```
EpicsTpy::multi_io_support::multi_io_support (
    const std::stringcase & dname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

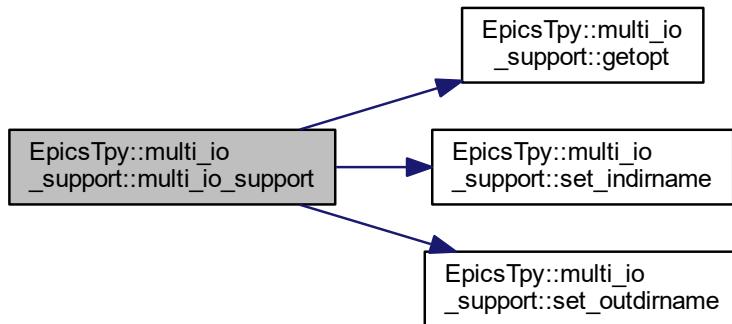
### Parameters

|              |  |
|--------------|--|
| <i>dname</i> | Name of output directory   |
| <i>argc</i>  | Number of command line arguments   |
| <i>argv</i>  | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i>  | Excluded/processed arguments (in/out), array length must be argc         |

Definition at line 340 of file TpyToEpics.h.

References EpicsTpy::closed, getopt(), set\_indirname(), and set\_outdirname().

Here is the call graph for this function:



### 8.36.3 Member Function Documentation

#### 8.36.3.1 getopt()

```

int EpicsTpy::multi_io_support::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline], [protected]
  
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

**Parameters**

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

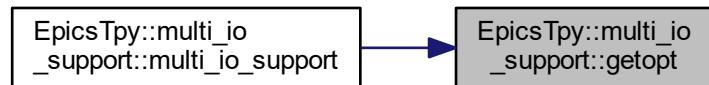
**Returns**

Number of arguments processed

Definition at line 399 of file TpyToEpics.h.

Referenced by [multi\\_io\\_support\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

## 8.37 ParseUtil::opc\_list Class Reference

OPC list.

```
#include <ParseUtil.h>
```

### Public Member Functions

- [`opc\_list \(\)`](#)  
*Default constructor.*
- [`opc\_list \(opc\_enum state, const property\_map &map\)`](#)  
*Constructor.*
- [`opc\_enum get\_opc\_state \(\) const`](#)  
*Get opc state.*
- [`void set\_opc\_state \(opc\_enum state\)`](#)  
*Set opc state.*
- [`const property\_map & get\_properties \(\) const`](#)

- `property_map & get_properties ()`

*Get opc property list.*
- `const property_map::const_iterator get_property (int key) const`

*Get the specified property.*
- `void add (const property_el &el)`

*Add an OPC property.*
- `void add (const opc_list &o)`

*Add an OPC list.*
- `bool is_published () const`

*Is this item published?*
- `bool is_READONLY () const`

*Is readonly?*
- `bool get_property (int prop, std::stringcase &val) const`

*Get string property.*
- `bool get_property (int prop, int &val) const`

*Get integer property.*
- `bool get_property (int prop, double &val) const`

*Get real property.*

## Protected Attributes

- `opc_enum opc`

*OPC state.*
- `property_map opc_prop`

*List of OPC properties.*

### 8.37.1 Detailed Description

OPC list.

This class stores OPC properties.

Definition at line 144 of file ParseUtil.h.

The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

## 8.38 ParseUtil::optarg Class Reference

Optional arguments.

```
#include <ParseUtil.h>
```

## Public Member Functions

- `optarg ()`  
*Default constructor.*
- `optarg (const std::stringcase &arg)`
- `~optarg ()`  
*Destructor.*
- `int parse (const std::stringcase &arg)`
- `int argc () const`  
*Returns the number of arguments.*
- `const char *const * argv () const`  
*Returns the argument list.*
- `const bool * argp () const`  
*Return the processed argument list.*
- `bool * argp ()`  
*Return the processed argument list.*
- `bool all_done () const`  
*Returns true if there all arguments are unprocessed.*

## Protected Member Functions

- `optarg (const optarg &)`  
*Disabled copy constructor.*
- `optarg & operator= (const optarg &)`  
*Disabled assignment operator.*
- `void setup (int size)`  
*Setup.*

## Protected Attributes

- `int mysize`  
*Size of allocated arrays.*
- `int myargc`  
*Number of arguments.*
- `char ** myargv`  
*Argument list.*
- `bool * myargp`  
*Processed argument list.*

### 8.38.1 Detailed Description

Optional arguments.

This class transforms a string into a standard program argument.

Definition at line 73 of file ParseUtil.h.

## 8.38.2 Constructor & Destructor Documentation

### 8.38.2.1 optarg()

```
ParseUtil::optarg::optarg (
    const std::string& arg ) [inline], [explicit]
```

Constructor

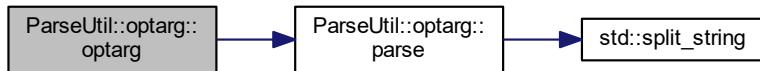
Parameters

|            |                        |
|------------|------------------------|
| <i>arg</i> | Option argument string |
|------------|------------------------|

Definition at line 80 of file ParseUtil.h.

References parse().

Here is the call graph for this function:



## 8.38.3 Member Function Documentation

### 8.38.3.1 parse()

```
int ParseUtil::optarg::parse (
    const std::string& arg )
```

Parse string argument

Parameters

|            |                        |
|------------|------------------------|
| <i>arg</i> | Option argument string |
|------------|------------------------|

Returns

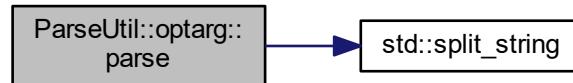
Number of processed arguments

Definition at line 60 of file ParseUtil.cpp.

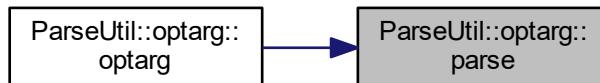
References std::split\_string().

Referenced by optarg().

Here is the call graph for this function:



Here is the caller graph for this function:



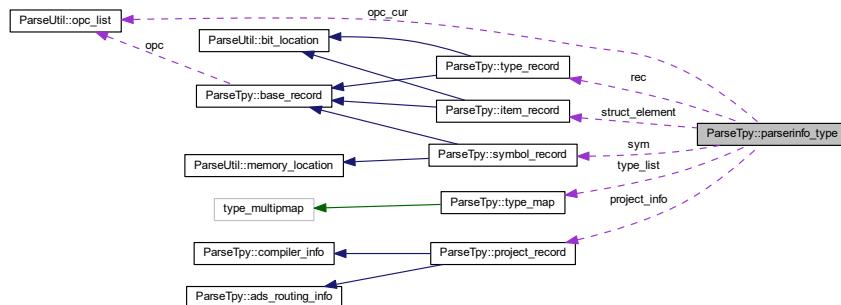
The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

## 8.39 ParseTpy::parserinfo\_type Class Reference

Parser information.

Collaboration diagram for ParseTpy::parserinfo\_type:



## Public Member Functions

- `parserinfo_type (project_record &p, symbol_list &s, type_map &t)`  
`Constructor.`
- `symbol_list & get_symbols ()`  
`Get symbol list.`
- `type_map & get_types ()`  
`Get type list.`
- `project_record & get_projectinfo ()`  
`Get project information.`
- `void init ()`  
`Initialize temporary parser info.`
- `type_enum get_type_description () const`  
`Get type of parsed object.`
- `bool verytop ()`  
`the very top of the xml tag hierarchy (not within any tag)`
- `bool top ()`  
`the top of the xml tag hierarchy (within the PlcProjectInfo tag)`

## Public Attributes

- `int ignore`  
`ignore elements during parsing (with level)`
- `bool projects`  
`parsing withing PlcProjectInfo tag`
- `int routing`  
`parsing within Routing tag (1) or AdsInfo (2), Net ID (3), Port (4), Target name (5)`
- `int compiler`  
`parsing within compiler tag (1) or compiler version (2), Twincat version (3), CPU family (4)`
- `int types`  
`parsing within DataTypes tag (1) or DataType tag (2)`
- `int symbols`  
`parsing within Symbols tag (1) or Symbol tag (2)`
- `symbol_record sym`  
`temporary symbol information during parsing`
- `type_record rec`  
`temporary type information during parsing`
- `int name_parse`
- `int type_parse`  
`level indicator for type parsing`
- `opc_list * opc_cur`  
`Pointer to current opc list (types only)`
- `property_el opc_prop`  
`temporary opc element`
- `int opc_parse`  
`level indicator for opc element`
- `std::stringcase opc_data`  
`temporary data string for parsed opc data`
- `int opc_cdata`  
`temporary cdata indicator`

- int `igroup_parse`  
*level indicator for IGroup*
- int `ioffset_parse`  
*level indicator for IOffset*
- int `bitsize_parse`  
*level indicator for BitSize*
- int `bitoffs_parse`  
*level indicator for BitOffs*
- `std::stringcase data`  
*temporary data string for parsed igroup/ioffset/bitsize/bitoffs*
- int `array_parse`  
*level indicator for array info parsing*
- `std::stringcase array_data`  
*temporary data string for parsed array info*
- `dimension array_bounds`  
*temporary array dimension element*
- int `enum_parse`  
*level indicator for enum parsing*
- `std::stringcase enum_data`  
*temporary data string for parsed enum data*
- `enum_pair enum_element`  
*temporary enum element*
- `std::stringcase enum_comment`  
*temporary enum comment*
- int `struct_parse`  
*level indicator for struct parsing*
- `item_record struct_element`  
*temporary structure element*
- int `fb_parse`  
*level indicator for function block parsing*

## Protected Attributes

- `symbol_list * sym_list`  
*pointer to symbol list*
- `type_map * type_list`  
*pointer to type list*
- `project_record * project_info`  
*pointer to project info*

### 8.39.1 Detailed Description

Parser information.

This structure keeps track of the parser information.

Definition at line 41 of file ParseTpy.cpp.

## 8.39.2 Member Data Documentation

### 8.39.2.1 name\_parse

```
int ParseTpy::parserinfo_type::name_parse
```

level indicator for type name parsing level indicators in general: 0 - not encountered, 1 - parsed, 2 - currently processing, 3 - further sub tag parsing

Definition at line 97 of file ParseTpy.cpp.

The documentation for this class was generated from the following file:

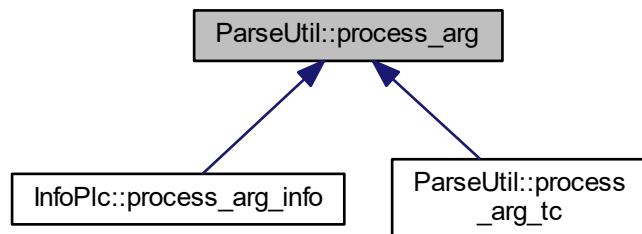
- [ParseTpy.cpp](#)

## 8.40 ParseUtil::process\_arg Class Reference

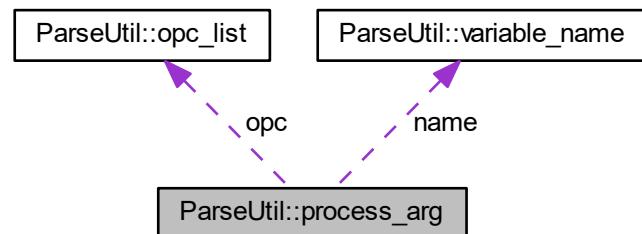
Arguments for processing.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::process\_arg:



Collaboration diagram for ParseUtil::process\_arg:



## Public Member Functions

- `process_arg (const variable_name &vname, process_type_enum pt, const opc_list &o, const std::stringcase &tname, bool at)`  
*Get variable.*
- `const variable_name & get_var () const`
- `const std::stringcase & get_name () const`  
*Get name.*
- `const std::stringcase & get_alias () const`  
*Get alias.*
- `const std::stringcase & get_type_name () const`  
*Get type name.*
- `const opc_list & get_opc () const`  
*Get OPC list.*
- `process_type_enum get_process_type () const`  
*Get process type.*
- `std::stringcase get_process_string () const`  
*Get process type string.*
- `bool is_atomic () const`  
*Is atomic (or structured) type.*
- `virtual std::stringcase get_full () const =0`

## Protected Attributes

- `const variable_name & name`  
*name of type*
- `const std::stringcase & type_n`  
*type definition*
- `const opc_list & opc`  
*list of opc properties*
- `process_type_enum ptype`  
*Process Type.*
- `bool atomic`  
*Atomic element.*

### 8.40.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 343 of file ParseUtil.h.

### 8.40.2 Constructor & Destructor Documentation

#### 8.40.2.1 process\_arg()

```
ParseUtil::process_arg::process_arg (
    const variable_name & vname,
    process_type_enum pt,
    const opc_list & o,
    const std::stringcase & tname,
    bool at ) [inline]
```

Constructor

**Parameters**

|              |               |
|--------------|---------------|
| <i>vname</i> | Variable name |
| <i>pt</i>    | Process type  |
| <i>o</i>     | OPC list      |
| <i>tname</i> | Type name     |
| <i>at</i>    | Atomic type   |

Definition at line 352 of file ParseUtil.h.

### 8.40.3 Member Function Documentation

#### 8.40.3.1 get\_full()

```
virtual std::string ParseUtil::process_arg::get_full() const [pure virtual]
```

Gets a string representation of a PLC & memory location

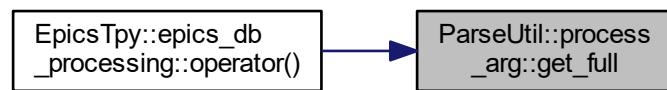
**Returns**

string with format "prefix:group/offset:size", empty on error

Implemented in [ParseUtil::process\\_arg\\_tc](#), and [InfoPlc::process\\_arg\\_info](#).

Referenced by [EpicsTpy::epics\\_db\\_processing::operator\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

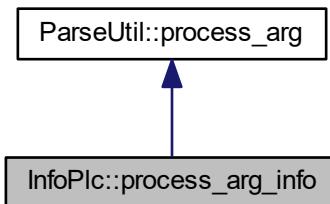
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

## 8.41 InfoPlc::process\_arg\_info Class Reference

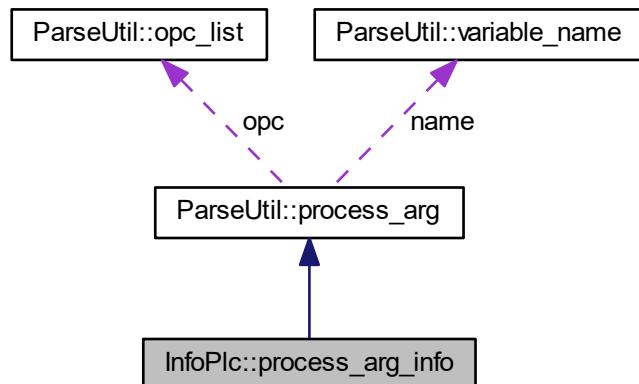
Arguments for processing.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::process\_arg\_info:



Collaboration diagram for InfoPlc::process\_arg\_info:



### Public Member Functions

- `process_arg_info (const ParseUtil::variable_name &vname, ParseUtil::process_type_enum pt, const ParseUtil::opc_list &o, const std::stringcase &tname, bool at, const std::stringcase &plcaddr)`
- `virtual std::stringcase get_full () const`

### Protected Attributes

- `const std::stringcase & tcplc_addr`  
*TwinCAT PLC address.*

### 8.41.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 218 of file infoPlc.h.

### 8.41.2 Constructor & Destructor Documentation

#### 8.41.2.1 process\_arg\_info()

```
InfoPlc::process_arg_info::process_arg_info (
    const ParseUtil::variable_name & vname,
    ParseUtil::process_type_enum pt,
    const ParseUtil::opc_list & o,
    const std::stringcase & tname,
    bool at,
    const std::stringcase & plcaddr ) [inline]
```

Constructor

##### Parameters

|         |                     |
|---------|---------------------|
| vname   | Variable name       |
| pt      | Process type        |
| o       | OPC list            |
| tname   | Type name           |
| at      | Atomic type         |
| plcaddr | TwinCAT PLC address |

Definition at line 228 of file infoPlc.h.

### 8.41.3 Member Function Documentation

#### 8.41.3.1 get\_full()

```
std::stringcase InfoPlc::process_arg_info::get_full () const [virtual]
```

Gets a string representation of a PLC & memory location

**Returns**

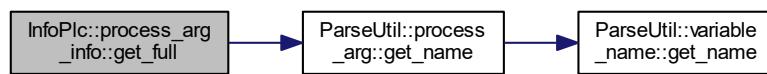
string with format "prefixigroup/offset:size", empty on error

Implements [ParseUtil::process\\_arg](#).

Definition at line 1231 of file [infoPlc.cpp](#).

References [ParseUtil::process\\_arg::get\\_name\(\)](#), and [tcplc\\_addr](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

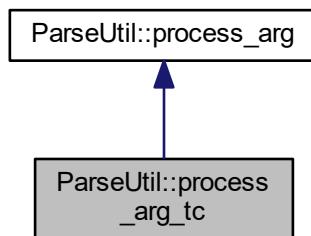
- [infoPlc.h](#)
- [infoPlc.cpp](#)

## 8.42 ParseUtil::process\_arg\_tc Class Reference

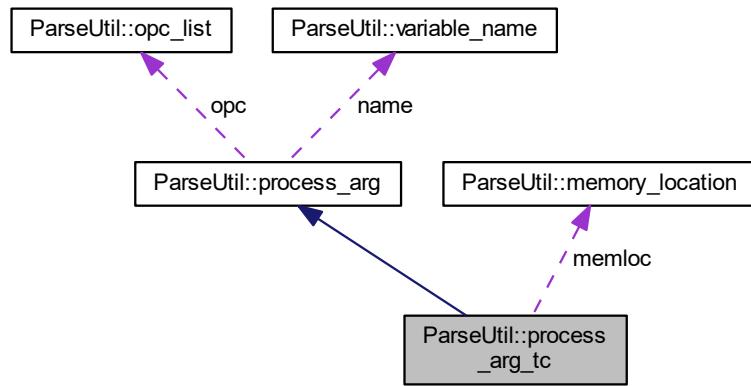
Arguments for processing.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::process\_arg\_tc:



Collaboration diagram for ParseUtil::process\_arg\_tc:



## Public Member Functions

- `process_arg_tc (const memory_location &loc, const variable_name &vname, process_type_enum pt, const opc_list &o, const std::stringcase &tname, bool at)`
- `int get_igroup () const`  
*Get IGroup.*
- `int get_ioffset () const`  
*Get IOffset.*
- `int get_bytesize () const`  
*Get BitSize.*
- `std::stringcase get () const`
- `virtual std::stringcase get_full () const`

## Protected Attributes

- `const memory_location & memloc`  
*memory location*

### 8.42.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 395 of file ParseUtil.h.

### 8.42.2 Constructor & Destructor Documentation

### 8.42.2.1 process\_arg\_tc()

```
ParseUtil::process_arg_tc::process_arg_tc (
    const memory_location & loc,
    const variable_name & vname,
    process_type_enum pt,
    const opc_list & o,
    const std::stringcase & tname,
    bool at ) [inline]
```

Constructor

Parameters

|              |                 |
|--------------|-----------------|
| <i>loc</i>   | Memory location |
| <i>vname</i> | Variable name   |
| <i>pt</i>    | Process type    |
| <i>o</i>     | OPC list        |
| <i>tname</i> | Type name       |
| <i>at</i>    | Atomic type     |

Definition at line 405 of file ParseUtil.h.

### 8.42.3 Member Function Documentation

#### 8.42.3.1 get()

```
std::stringcase ParseUtil::process_arg_tc::get () const [inline]
```

Gets a string representation of a memory location

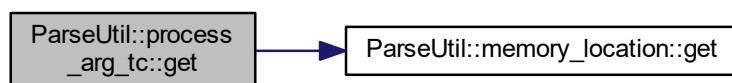
Returns

string with format "igroup/offset:size", empty on error

Definition at line 418 of file ParseUtil.h.

References ParseUtil::memory\_location::get(), and memloc.

Here is the call graph for this function:



### 8.42.3.2 get\_full()

```
std::string ParseUtil::process_arg_tc::get_full () const [virtual]
```

Gets a string representation of a PLC & memory location

Returns

string with format "prefix:group/offset:size", empty on error

Implements [ParseUtil::process\\_arg](#).

Definition at line 322 of file ParseUtil.cpp.

References ParseUtil::OPC\_PROP\_PLCNAME.

The documentation for this class was generated from the following files:

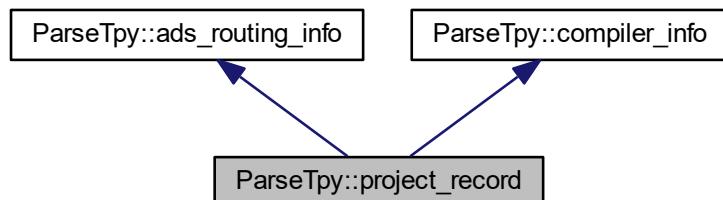
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

## 8.43 ParseTpy::project\_record Class Reference

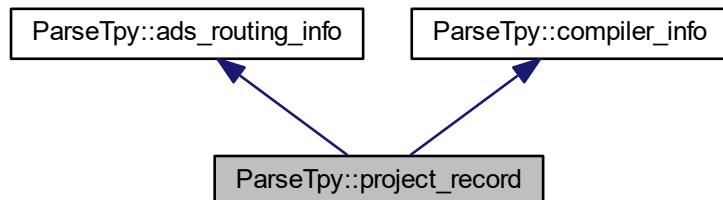
Project information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::project\_record:



Collaboration diagram for ParseTpy::project\_record:



## Public Member Functions

- [project\\_record \(\)](#)

*Default constructor.*

## Additional Inherited Members

### 8.43.1 Detailed Description

Project information.

This is a base class for storing the project information

Definition at line 137 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

## 8.44 DevTc::register\_devsup Class Reference

Device support registration.

```
#include <devTc.h>
```

Collaboration diagram for DevTc::register\_devsup:

DevTc::register\_devsup ↗ the\_register\_devsup

## Public Types

- [typedef auto link\\_func\(dbCommon \\*pEpicsRecord, plc::BaseRecordPtr &pRecord\) -> bool](#)  
*Type describing the link function.*
- [typedef std::pair< std::regex, link\\_func & > test\\_pattern](#)  
*pair of pattern and link function*
- [typedef std::vector< test\\_pattern > test\\_pattern\\_list](#)  
*list of pattern/link functions*

## Static Public Member Functions

- static void [add](#) (const std::regex &rgx, [link\\_func](#) &func)  
*Register a pattern/link function.*
- static bool [linkRecord](#) (const std::stringcase &inpout, dbCommon \*pEpicsRecord, [plc::BaseRecordPtr](#) &p← Record)  
*linkRecord*

## Protected Member Functions

- [register\\_devsup](#) ()  
*Default constructor (adds linkTcRecord entry)*
- [register\\_devsup](#) (const [register\\_devsup](#) &)  
*Disabled copy constructor.*
- [register\\_devsup](#) & [operator=](#) (const [register\\_devsup](#) &)  
*Disabled assignment operator.*

## Protected Attributes

- [test\\_pattern\\_list tp\\_list](#)  
*list of pattern and links*

## Static Protected Attributes

- static [register\\_devsup the\\_register\\_devsup](#)  
*the one global instance of the register class*

### 8.44.1 Detailed Description

Device support registration.

This is a class for managing device support for multiple record types, such as TwinCAT/ADS and Info.

Definition at line 49 of file devTc.h.

The documentation for this class was generated from the following files:

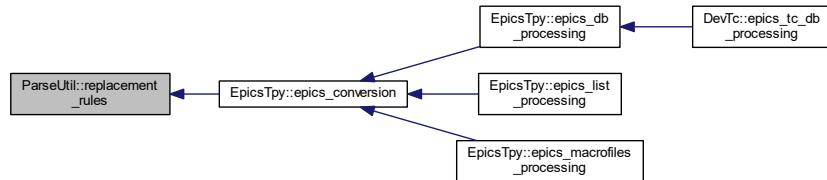
- [devTc.h](#)
- [devTc.cpp](#)

## 8.45 ParseUtil::replacement\_rules Class Reference

Replacement rules.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::replacement\_rules:



### Public Member Functions

- `replacement_rules` (bool rec=true)  
*Default constructor.*
- `replacement_rules` (const `replacement_table` &t, bool rec=true)  
*Constructor.*
- `void add_rule` (const `std::stringcase` &var, const `std::stringcase` &val)  
*Add a rule.*
- `void set_rule_table` (const `replacement_table` &t)  
*set table*
- `replacement_table & get_rule_table` ()  
*get table*
- `const replacement_table & get_rule_table` () const  
*get table*
- `std::stringcase apply_replacement_rules` (const `std::stringcase` &s) const  
*replace*
- `bool HasRules` () const  
*Has rules.*
- `bool is_recursive` () const  
*Is recursive?*
- `void set_recursive` (bool rec)  
*Set recursive.*

### Static Public Attributes

- `static const char *const prefix = "${"`  
*prefix for replacement rule: \${*
- `static const char *const suffix = "}"`  
*suffix for replacement rule: }*

## Protected Attributes

- [replacement\\_table](#) *table*  
*Replacement table.*
- [bool recursive](#)  
*Recursive replacement.*

### 8.45.1 Detailed Description

Replacement rules.

Epics channel conversion arguments Epics channels are generated from opc through a conversion rule

Definition at line 27 of file ParseUtil.h.

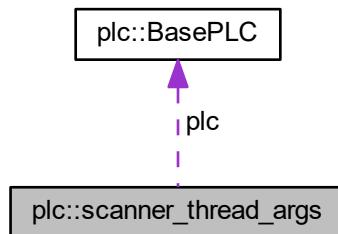
The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

## 8.46 plc::scanner\_thread\_args Struct Reference

Scanner thread arguments.

Collaboration diagram for plc::scanner\_thread\_args:



## Public Attributes

- [plc::BasePLC \\* plc](#)  
*PLC this scanner operates on.*
- [long scanperiod](#)  
*Period in ms of the scanner.*
- [plc::BasePLC::scanner\\_func scanner](#)  
*Address of the scanner function to be used (read, write, or update)*

### 8.46.1 Detailed Description

Scanner thread arguments.

Structure for arguments sent to a scanner thread

Definition at line 662 of file plcBase.cpp.

The documentation for this struct was generated from the following file:

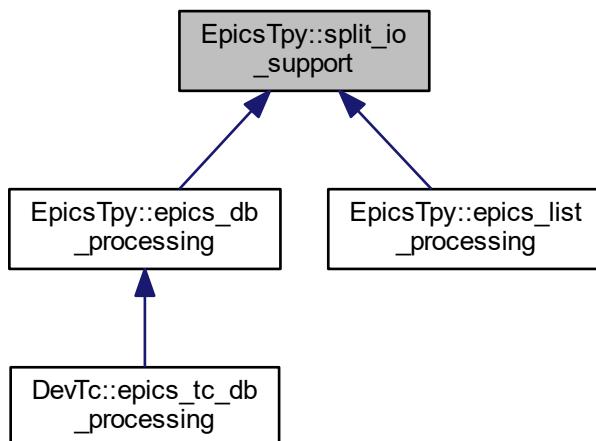
- [plcBase.cpp](#)

## 8.47 EpicsTpy::split\_io\_support Class Reference

Split IO support.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::split\_io\_support:



### Public Member Functions

- [split\\_io\\_support \(\)](#)  
*Default constructor.*
- [split\\_io\\_support \(const std::string& fname, bool split=false, int max=0\)](#)  
*Constructor.*
- [split\\_io\\_support \(const std::string& fname, int argc, const char \\*const argv\[\], bool argp\[\] = 0\)](#)
- [~split\\_io\\_support \(\)](#)  
*Destructor.*
- [split\\_io\\_support \(const split\\_io\\_support &\)](#)

- `split_io_support & operator= (const split_io_support &)`
- `bool operator! () const`

*Return error.*
- `bool increment (bool readonly)`
- `void flush ()`

*Flush contents of output files.*
- `FILE * get_file () const`

*Get output file.*
- `const std::stringcase & get_filename () const`

*Get output filename.*
- `bool is_split () const`

*Is output split?*
- `int get_max () const`

*Maximum of channels per file.*
- `int get_processed_total () const`

*Get number of processed channels.*
- `int get_processed_READONLY () const`

*Get number of processed readonly channels.*
- `int get_processed_IO () const`

*Get number of processed input/output channels.*

## Protected Member Functions

- `void set_filename (const std::stringcase &fname)`

*Set output filename.*
- `void close ()`

*Close files.*
- `void set_split (bool split)`

*set split*
- `void set_max (int max)`

*Set maximum of channels per file.*
- `int getopt (int argc, const char *const argv[], bool argp[] = 0)`

## Protected Attributes

- `bool error`

*Error.*
- `std::stringcase outfilename`

*Output filename.*
- `bool split_io`

*Split output into read only channels and input/output channels.*
- `int split_n`

*Maximum number of channels per file; 0 indicates no limit.*
- `FILE * outf`

*Output file.*
- `FILE * outf_in`

*Output file for read only channels.*
- `FILE * outf_io`

*Output file for input/output channels.*

- int `rec_num`  
*Current number of processed channels (records)*
- int `rec_num_in`  
*Current number of processed read only channels (records)*
- int `rec_num_io`  
*Current number of processed input/output channels (records)*
- int `file_num_in`  
*Current file number of processed read only channels (records)*
- int `file_num_io`  
*Current file number of processed input/output channels (records)*
- `std::stringcase file_num_in_s`  
*Contains the readonly file number in string format.*
- `std::stringcase file_num_io_s`  
*Contains the input/output file number in string format.*
- `std::stringcase file_in_s`  
*Contains the file extenstion for readonly files.*
- `std::stringcase file_io_s`  
*Contains the file extenstion for input/output files.*

### 8.47.1 Detailed Description

Split IO support.

Split file IO support Output can be split in multiple files if the number of channels exceeds the maximum specified for a file

Definition at line 170 of file TpyToEpics.h.

### 8.47.2 Constructor & Destructor Documentation

#### 8.47.2.1 `split_io_support()` [1/2]

```
EpicsTpy::split_io_support::split_io_support (
    const std::stringcase & fname,
    int argc,
    const char *const argv[],
    bool argp[ ] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

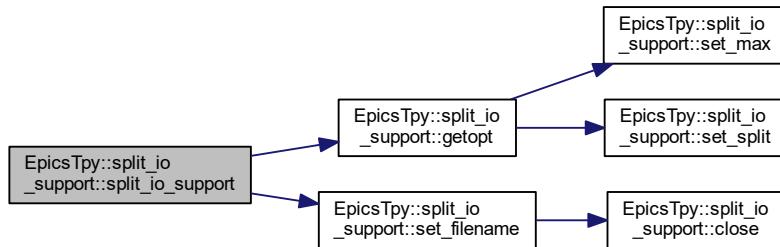
#### Parameters

|                    |   |
|--------------------|---|
| <code>fname</code> | Filename for output   |
| <code>argc</code>  | Number of command line arguments                                      |
| <code>argv</code>  | List of command line arguments, same format as in <code>main()</code> |
| <code>argp</code>  | Excluded/processed arguments (in/out), array length must be argc      |

Definition at line 195 of file TpyToEpics.h.

References getopt(), and set\_filename().

Here is the call graph for this function:



#### 8.47.2.2 split\_io\_support() [2/2]

```
EpicsTpy::split_io_support::split_io_support (
    const split_io_support & iosup )
```

Copy constructor File pointers will be moved over and the original ones will become invalid.

Definition at line 209 of file TpyToEpics.cpp.

### 8.47.3 Member Function Documentation

#### 8.47.3.1 getopt()

```
int EpicsTpy::split_io_support::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [protected]
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/ysio: Splits database into input only and input/output records /nsio: Does not split database by record type (default) /sn 'num': Splits database or listing into files with no more than num records /sn 0: Does not split database or listing into multiple files (default)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

**Parameters**

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

**Returns**

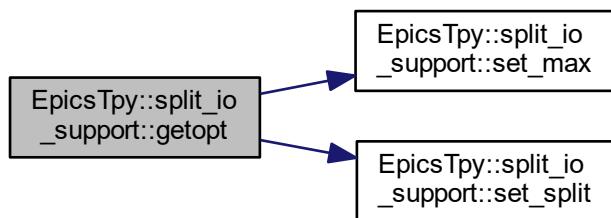
Number of arguments processed

Definition at line 247 of file TpyToEpics.cpp.

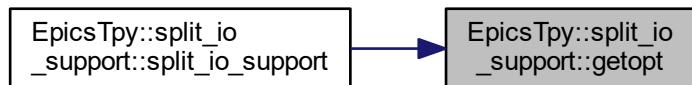
References [set\\_max\(\)](#), and [set\\_split\(\)](#).

Referenced by [split\\_io\\_support\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**8.47.3.2 increment()**

```
bool EpicsTpy::split_io_support::increment (
    bool readonly )
```

Increase the channel number

**Parameters**

|                       |                                  |
|-----------------------|----------------------------------|
| <code>readonly</code> | Indicates if channel is readonly |
|-----------------------|----------------------------------|

**Returns**

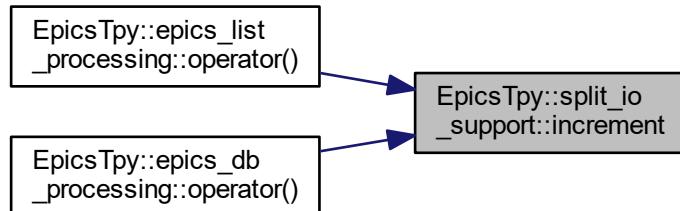
True if no error

Definition at line 383 of file TpyToEpics.cpp.

References error, file\_in\_s, file\_io\_s, file\_num\_in, file\_num\_in\_s, file\_num\_io, file\_num\_io\_s, outf, outf\_in, outf\_io, outfilename, rec\_num, rec\_num\_in, rec\_num\_io, split\_io, and split\_n.

Referenced by EpicsTpy::epics\_list\_processing::operator()(), and EpicsTpy::epics\_db\_processing::operator()().

Here is the caller graph for this function:

**8.47.3.3 operator=( )**

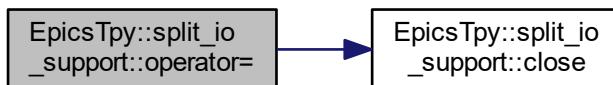
```
split_io_support & EpicsTpy::split_io_support::operator= (
    const split_io_support & iosup )
```

Assignment File pointers will be moved over and the original ones will become invalid.

Definition at line 220 of file TpyToEpics.cpp.

References close(), error, file\_in\_s, file\_io\_s, file\_num\_in, file\_num\_in\_s, file\_num\_io, file\_num\_io\_s, outf, outf\_in, outf\_io, outfilename, rec\_num, rec\_num\_in, rec\_num\_io, split\_io, and split\_n.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

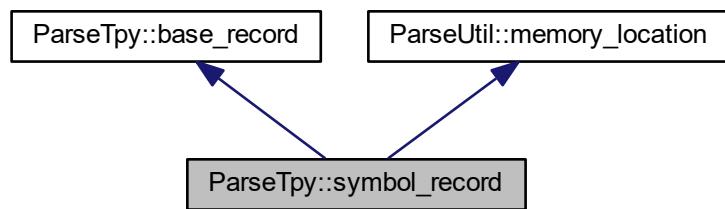
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

## 8.48 ParseTpy::symbol\_record Class Reference

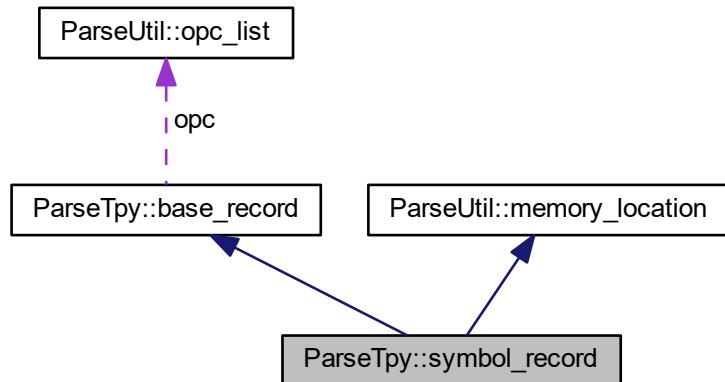
Symbol record.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::symbol\_record:



Collaboration diagram for ParseTpy::symbol\_record:



### Public Member Functions

- [symbol\\_record \(\)](#)  
*Default constructor.*
- [const memory\\_location & get\\_location \(\) const](#)  
*get memory location*
- [memory\\_location & get\\_location \(\)](#)  
*get memory location*

## Additional Inherited Members

### 8.48.1 Detailed Description

Symbol record.

This structure holds a symbol record

Definition at line 340 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

## 8.49 syminfo\_processing Class Reference

Symbol processing.

### Public Member Functions

- [`syminfo\_processing`](#) (FILE \*outfile=0, bool atomic=true)  
*Constructor.*
- [`bool operator\(\)`](#) (const [process\\_arg](#) &arg)  
*Process.*

### Protected Attributes

- [`FILE \* outf`](#)  
*Ouptut file.*
- [`bool firstline`](#)  
*Firstline?*

### 8.49.1 Detailed Description

Symbol processing.

Symbol processing

Definition at line 16 of file ParseTpyInfo.cpp.

The documentation for this class was generated from the following file:

- [ParseTpyInfo.cpp](#)

## 8.50 plc::System Class Reference

[System](#) to keep track of PLCs.

```
#include <plcBase.h>
```

### Public Types

- `typedef std::recursive_mutex mutex_type`  
*Defines the mutex type.*
- `typedef std::lock_guard< mutex_type > guard`  
*Defined the mutex guard type.*

### Public Member Functions

- `bool add (BasePLC *plc)`
- `bool add (BasePLCPtr plc)`
- `BasePLCPtr find (std::stringcase id)`  
*Finds a PLC by its name.*
- template<typename func >  
`void for_each (func &f)`
- `void printVals ()`  
*Print all PLC record values to the console.*
- `void printVal (const std::string &var)`
- `void start ()`  
*Start scanning after ioc is running.*
- `void stop ()`  
*Stop scanning when ioc is paused.*
- `bool is_ioc_running () const`  
*get loc run state*
- `void set_ioc_state (bool run)`  
*set loc run state*

### Static Public Member Functions

- `static System & get ()`  
*Return a reference to the gloabl System variable.*

### Protected Attributes

- `mutex_type mux`  
*Mutex to synchronize access to this class.*
- `BasePLCList PLCs`  
*Master list of all PLCs.*
- `bool locRun`  
*IOC is running.*

### 8.50.1 Detailed Description

[System](#) to keep track of PLCs.

This is a class for managing multiple PLCs.

Definition at line 899 of file plcBase.h.

### 8.50.2 Member Function Documentation

#### 8.50.2.1 add() [1/2]

```
bool plc::System::add (
    BasePLC * plc ) [inline]
```

Add a new PLC, PLC will be adopted

##### Parameters

|            |                |
|------------|----------------|
| <i>plc</i> | Pointer to plc |
|------------|----------------|

##### Returns

true if successful

Definition at line 912 of file plcBase.h.

#### 8.50.2.2 add() [2/2]

```
bool plc::System::add (
    BasePLCPtr plc )
```

Add a new PLC

##### Parameters

|            |                      |
|------------|----------------------|
| <i>plc</i> | Smart pointer to plc |
|------------|----------------------|

##### Returns

true if successful

Definition at line 788 of file plcBase.cpp.

References mux, and PLCs.

### 8.50.2.3 `for_each()`

```
template<typename func >
void plc::System::for_each (
    func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

#### Parameters

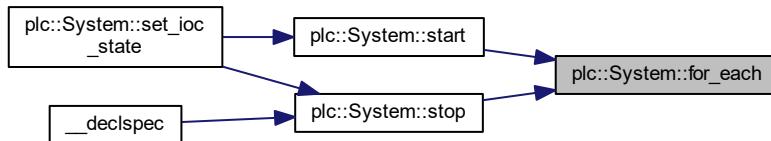
|                |   |
|----------------|---|
| <code>f</code> | Function which takes <code>BaseRecord*</code> as the argument |
|----------------|---|

Definition at line 216 of file `plcBaseTemplate.h`.

References mux, and PLCs.

Referenced by `start()`, and `stop()`.

Here is the caller graph for this function:



### 8.50.2.4 `printVal()`

```
void plc::System::printVal (
    const std::string & var )
```

Print PLC record value to the console

#### Parameters

|                  |                           |
|------------------|---------------------------|
| <code>var</code> | Variable name or wildcard |
|------------------|---------------------------|

Definition at line 813 of file `plcBase.cpp`.

References mux, and PLCs.

Referenced by DevTc::tcPrintVal().

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

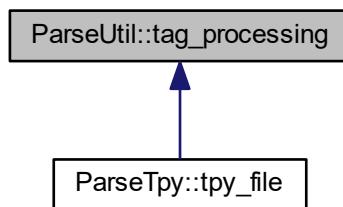
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

## 8.51 ParseUtil::tag\_processing Class Reference

Tag processing selection.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::tag\_processing:



### Public Member Functions

- [tag\\_processing \(\)](#)  
*Default constructor.*
- [tag\\_processing \(bool all, process\\_tag\\_enum proctags, bool nostring=false\)](#)
- [tag\\_processing \(int argc, const char \\*const argv\[\], bool argp\[\] = 0\)](#)
- [int getopt \(int argc, const char \\*const argv\[\], bool argp\[\] = 0\)](#)
- [bool get\\_export\\_all \(\) const](#)  
*Get the export rule.*

- void `set_export_all` (bool all)  
*Set the export rule.*
- `process_tag_enum get_process_tags () const`  
*Get the process rule.*
- void `set_process_tags` (`process_tag_enum` proctags)  
*Set the process rule.*
- bool `get_no_strings () const`  
*Get the string rule.*
- void `set_no_strings` (bool nostring)  
*Set the string rule.*

## Protected Attributes

- bool `export_all`  
*Process all symbols regardless of opc publish setting.*
- `process_tag_enum process_tags`  
*Process only atomic types.*
- bool `no_string_tags`  
*Don't process strings.*

### 8.51.1 Detailed Description

Tag processing selection.

Class to specify which symbols and tags/names to process

Definition at line 444 of file ParseUtil.h.

### 8.51.2 Constructor & Destructor Documentation

#### 8.51.2.1 `tag_processing()` [1/2]

```
ParseUtil::tag_processing::tag_processing (
    bool all,
    process_tag_enum proctags,
    bool nostring = false )  [inline]
```

Constructor

Parameters

|                       |                                       |
|-----------------------|---------------------------------------|
| <code>all</code>      | Process all tags                      |
| <code>proctags</code> | Process atomic and/or structured tags |
| <code>nostring</code> | Don't process string tags             |

Definition at line 454 of file ParseUtil.h.

References ParseUtil::all.

### 8.51.2.2 tag\_processing() [2/2]

```
ParseUtil::tag_processing::tag_processing (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Commaline arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

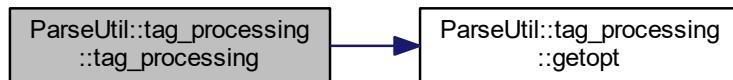
#### Parameters

|             |  |
|-------------|--|
| <i>argc</i> | Number of command line arguments   |
| <i>argv</i> | List of command line arguments, same format as in <a href="#">main()</a> |
| <i>argp</i> | Excluded/processed arguments (in/out), array length must be argc         |

Definition at line 465 of file ParseUtil.h.

References ParseUtil::all, and getopt().

Here is the call graph for this function:



### 8.51.3 Member Function Documentation

#### 8.51.3.1 getopt()

```
int ParseUtil::tag_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/ea: Export all variables regardless of OPC setting /eo: Only export variables which are marked as OPC export (default) /ns: No string variables are processed /ys: String variables are processes (default) /pa: Call process for all types (default) /ps: Call process for simple (atomic) types only /pc: Call process for complex (structure and array) types only

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

#### Parameters

|                   |  |
|-------------------|--|
| <code>argc</code> | Number of command line arguments   |
| <code>argv</code> | List of command line arguments, same format as in <a href="#">main()</a> |
| <code>argp</code> | Excluded/processed arguments (in/out), array length must be argc         |

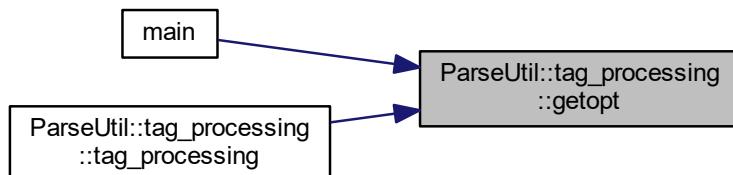
#### Returns

Number of arguments processed

Definition at line 338 of file ParseUtil.cpp.

Referenced by [main\(\)](#), and [tag\\_processing\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

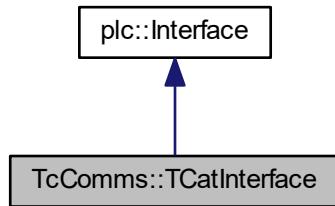
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

## 8.52 TcComms::TCatInterface Class Reference

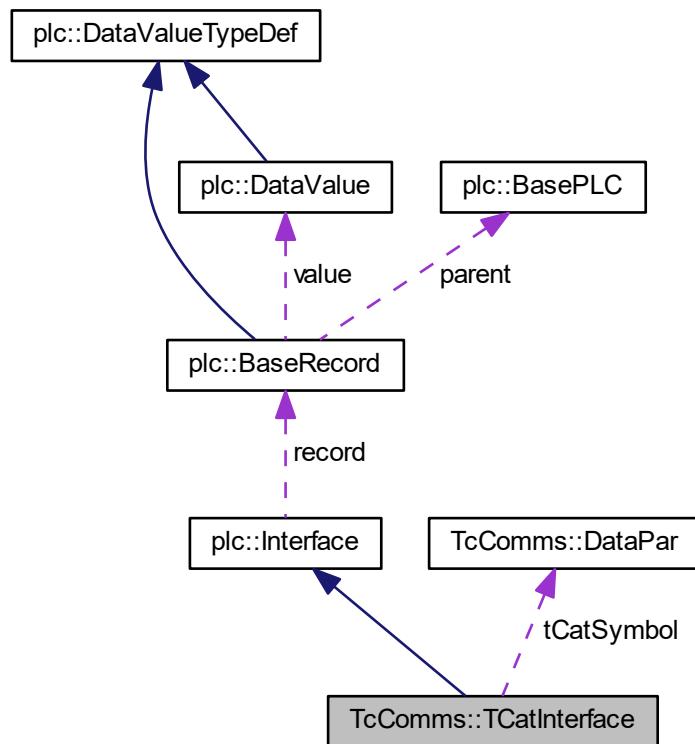
TCat interface class.

```
#include <tccomms.h>
```

Inheritance diagram for TcComms::TCatInterface:



Collaboration diagram for TcComms::TCatInterface:



## Public Member Functions

- `TCatInterface (plc::BaseRecord &dval)`  
`Constructor.`
- `TCatInterface (plc::BaseRecord &dval, const std::stringcase &name, unsigned long group, unsigned long offset, unsigned long length, const std::stringcase &type, bool isStruct, bool isEnum)`
- `~TCatInterface ()`  
`Deconstructor.`
- `const std::stringcase get_tCatName () const`  
`Get name of TCat symbol.`
- `void set_tCatName (std::stringcase name)`  
`Set name of TCat symbol.`
- `virtual const char * get_symbol_name () const`  
`Get symbol name.`
- `const std::stringcase get_tCatType ()`  
`Get TCat data type.`
- `void set_tCatType (std::stringcase type)`  
`Set TCat data type.`
- `DataPar get_tCatSymbol ()`  
`Get structure containing index group, index offset, size.`
- `unsigned long get_indexGroup () const`  
`Get index group.`
- `void set_indexGroup (unsigned long group)`  
`Set index group.`
- `unsigned long get_indexOffset () const`  
`Get index offset.`
- `void set_indexOffset (unsigned long offset)`  
`Set index offset.`
- `unsigned long get_size () const`  
`Get size in bytes of symbol.`
- `void set_size (unsigned long nBytes)`  
`Set size in bytes of symbol.`
- `size_t get_requestOffs () const`  
`Get offset into response buffer.`
- `void set_requestOffs (size_t pVal)`  
`Set offset into response buffer.`
- `TcPLC * get_parent ()`  
`Get parent PLC that owns this record.`
- `const TcPLC * get_parent () const`  
`Get parent PLC that owns this record.`
- `int get_requestNum ()`  
`Get the request group number this record is in.`
- `void set_requestNum (int rNum)`  
`Set the request group number this record is in.`
- `virtual void printVal (FILE *fp)`
- `virtual bool push () override`  
`Does nothing.`
- `virtual bool pull () override`  
`Does nothing.`

## Protected Attributes

- `std::stringcase tCatName`  
*Name of TCat symbol.*
- `std::stringcase tCatType`  
*Data type in TCat.*
- `DataPar tCatSymbol`  
*Struct storing index group, index offset, and length of TC symbol.*
- `int requestNum`  
*Which request group in the PLC.*
- `size_t requestOffs`  
*Offset into response buffer.*

### 8.52.1 Detailed Description

TCat interface class.

This is a class for a TCat interface

Definition at line 70 of file tcComms.h.

### 8.52.2 Constructor & Destructor Documentation

#### 8.52.2.1 TCatInterface()

```
TcComms::TCatInterface::TCatInterface (
    plc::BaseRecord & dval,
    const std::stringcase & name,
    unsigned long group,
    unsigned long offset,
    unsigned long length,
    const std::stringcase & type,
    bool isStruct,
    bool isEnum )
```

Constructor

#### Parameters

|                       |   |
|-----------------------|---|
| <code>dval</code>     | BaseRecord that this interface is part of |
| <code>name</code>     | Name of TCat symbol                       |
| <code>group</code>    | Index group of TCat symbol                |
| <code>offset</code>   | Index offset of TCat symbol               |
| <code>length</code>   | Size in bytes of TCat symbol              |
| <code>type</code>     | Name of TCat data type                    |
| <code>isStruct</code> | True = this symbol is a structure in TCat |
| <code>isEnum</code>   | True = this symbol is an enum in TCat     |

Definition at line 87 of file tcComms.cpp.

### 8.52.3 Member Function Documentation

#### 8.52.3.1 printVal()

```
void TcComms::TCatInterface::printVal (
    FILE * fp ) [virtual]
```

Prints TCat symbol value and information

##### Parameters

|           |                         |
|-----------|-------------------------|
| <i>fp</i> | File to print symbol to |
|-----------|-------------------------|

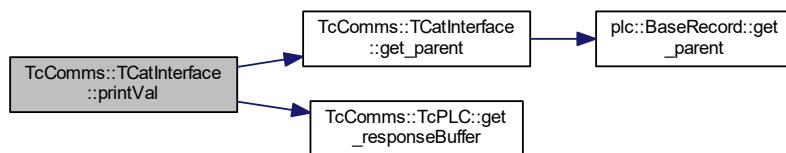
This is a function for printing the variable name and value of a record. Depending on the variable type, the readout from the ADS server is cast into the proper data type and printed to the output file fp.

Reimplemented from [plc::Interface](#).

Definition at line 131 of file tcComms.cpp.

References `get_parent()`, `TcComms::TcPLC::get_responseBuffer()`, `TcComms::DataPar::length`, `requestNum`, `requestOffs`, `tCatName`, `tCatSymbol`, and `tCatType`.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

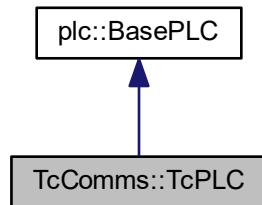
- [tcComms.h](#)
- [tcComms.cpp](#)

## 8.53 TcComms::TcPLC Class Reference

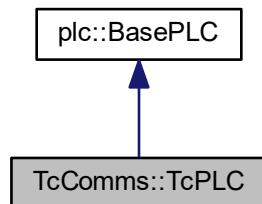
TwinCAT PLC.

```
#include <tcComms.h>
```

Inheritance diagram for TcComms::TcPLC:



Collaboration diagram for TcComms::TcPLC:



### Public Types

- `typedef char buffer_type`  
*Buffer type.*
- `typedef std::shared_ptr<buffer_type> buffer_ptr`  
*Smart pointer to buffer.*

### Public Member Functions

- `TcPLC (std::string tpyPath)`  
*Constructor.*
- `~TcPLC ()`  
*Destructor.*

- `bool is_valid_tpy ()`  
*Is typ still valid? Meaning, it hasn't changed.*
- `AmsAddr get_addr () const`  
*Get AMS netID of TwinCAT system and port number for this PLC.*
- `bool set_addr (std::string& netid, int port)`
- `long get_nReadPort () const`  
*Get read port number.*
- `long get_nWritePort () const`  
*Get write port number.*
- `int get_read_scanner_multiple () const`  
*Get slowdown multiple for EPICS read.*
- `void set_read_scanner_multiple (int mult)`  
*Set slowdown multiple for EPICS read.*
- `ADSSTATE get_ads_state () const`  
*Get ADS state.*
- `bool is_read_active () const`  
*Is read scanner active and successful.*
- `const std::string & get_tpyfilename () const`  
*Get the tpy filename.*
- `bool is_tpyfile_valid () const`  
*Is the tpy file valid?*
- `time_t get_tpyfile_time () const`  
*Get the file modification time of the tpy file.*
- `virtual bool start ()`  
*Starts the appropriate scanners.*
- `bool optimizeRequests ()`
- `buffer_ptr get_responseBuffer (size_t idx)`
- `virtual void printAllRecords ()`  
*Prints symbol information for entire list of symbols to console.*
- `virtual void printRecord (const std::string &var)`

## Protected Member Functions

- `virtual void read_scanner ()`  
*Makes read requests to ADS, makes PlcWrite on all data values.*
- `virtual void write_scanner ()`  
*Collects records to be written to TCat, makes write request.*
- `virtual void update_scanner ()`  
*Makes sure we don't have stale values.*
- `void set_ads_state (ADSSTATE state)`  
*Set ADS state.*
- `void setup_ads_notification ()`  
*Set up ADS status change notification.*
- `void remove_ads_notification ()`  
*Remove ADS status change notification.*
- `long openPort ()`  
*Opens a new ADS communication port.*
- `void closePort (long nPort)`

## Protected Attributes

- std::mutex `sync`  
*Mutex.*
- AmsAddr `addr`  
*AMS netID of TwinCAT system and port number for this PLC.*
- std::string `pathTpy`  
*The path of the tpy file.*
- time\_t `timeTpy`  
*Modification time of file.*
- std::atomic< bool > `checkTpy`  
*need to check modification time to make sure tpy file hasn't changed*
- bool `validTpy`  
*tpy file is valid and hasn't changed*
- int `nRequest`  
*Number of read request groups.*
- std::vector< DataPar > `adsGroupReadRequestVector`  
*Vector of index group, index offset, size for read requests.*
- std::vector< buffer\_ptr > `adsResponseBufferVector`  
*Vector of buffers for each read request group.*
- plc::BaseRecordList `nonTcRecords`  
*List of all records that don't interface directly with a PLC (info)*
- int `scanRateMultiple`  
*Slowdown multiple for EPICS read.*
- int `cyclesLeft`
- int `update_workload`  
*Workload for update scanner.*
- plc::BaseRecordPtr `update_last`  
*last updated record*
- std::atomic< ADSSTATE > `ads_state`  
*ADS state.*
- unsigned long `ads_handle`  
*ADS handle.*
- std::atomic< bool > `ads_restart`  
*ADS restart.*
- long `nReadPort`  
*Port number for ADS read connection.*
- long `nWritePort`  
*Port number for ADS write connection.*
- long `nNotificationPort`  
*Port number for ADS notification connection.*
- bool `read_active`  
*read active and successful*

## Friends

- void \_\_stdcall `ADScallback` (AmsAddr \*, AdsNotificationHeader \*, unsigned long)  
*Notification callback is a friend.*

### 8.53.1 Detailed Description

TwinCAT PLC.

Class for a connection to a TwinCAT PLC This class is derived from a BasePLC object, and specializes in managing records that contain a plc interface for TCat. This class is initialized using a .tpy file, from which it will obtain the AMS address information for connecting with TCat through ADS.

Reading and writing from/to ADS will be managed by this class, with read requests being grouped by continuous memory region in TCat to optimize read scanning for speed. Write requests are made using an ADS sum request.

There is also an option to send a request to ADS to check the status of both the PLC device and also the ADS connection.

Definition at line 257 of file tcComms.h.

### 8.53.2 Member Function Documentation

#### 8.53.2.1 closePort()

```
void TcComms::TcPLC::closePort (
    long nPort ) [protected]
```

Closes an ADS communication port

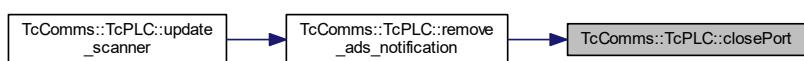
##### Parameters

|              |                         |
|--------------|-------------------------|
| <i>nPort</i> | Number of port to close |
|--------------|-------------------------|

Definition at line 864 of file tcComms.cpp.

Referenced by remove\_ads\_notification().

Here is the caller graph for this function:



#### 8.53.2.2 get\_responseBuffer()

```
TcPLC::buffer_ptr TcComms::TcPLC::get_responseBuffer (
    size_t idx )
```

Get pointer to the beginning of a read request response buffer

**Parameters**

|            |                          |
|------------|--------------------------|
| <i>idx</i> | Index of response buffer |
|------------|--------------------------|

**Returns**

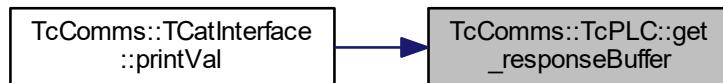
pointer to buffer

Definition at line 550 of file tcComms.cpp.

References adsResponseBufferVector.

Referenced by TcComms::TCatInterface::printVal().

Here is the caller graph for this function:

**8.53.2.3 optimizeRequests()**

```
bool TcComms::TcPLC::optimizeRequests ( )
```

Sorts read channels into request groups. Will make a new request group for channels not in continuous memory region in TCat. Will create buffers of appropriate size for each read request, and let each TCat record know where in the read response buffer the data for that symbol is.

**Returns**

true if successful

Definition at line 433 of file tcComms.cpp.

**8.53.2.4 printRecord()**

```
void TcComms::TcPLC::printRecord (
    const std::string & var ) [virtual]
```

Print a record values to stdout. (override for action)

**Parameters**

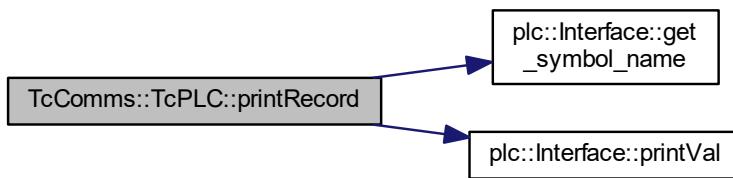
|                  |                                   |
|------------------|-----------------------------------|
| <code>var</code> | variable name (accepts wildcards) |
|------------------|-----------------------------------|

Reimplemented from [plc::BasePLC](#).

Definition at line 626 of file `tcComms.cpp`.

References `plc::Interface::get_symbol_name()`, `plc::Interface::printVal()`, and `plc::BasePLC::records`.

Here is the call graph for this function:

**8.53.2.5 set\_addr()**

```
bool TcComms::TcPLC::set_addr (
    std::stringcase netid,
    int port )
```

Set AMS address

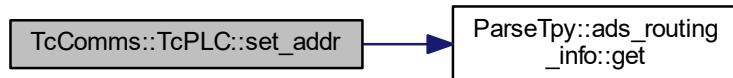
**Returns**

true if successful

Definition at line 342 of file `tcComms.cpp`.

References `addr`, and `ParseTpy::ads_routing_info::get()`.

Here is the call graph for this function:



### 8.53.2.6 set\_ads\_state()

```
void TcComms::TcPLC::set_ads_state (
    ADSSTATE state ) [protected]
```

Set ADS state.

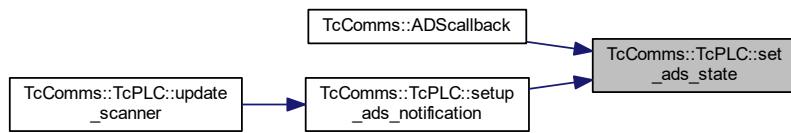
[TcPLC::set\\_ads\\_state](#)

Definition at line 675 of file tcComms.cpp.

References ads\_state, checkTpy, and plc::BasePLC::name.

Referenced by TcComms::ADSCallback(), and setup\_ads\_notification().

Here is the caller graph for this function:



### 8.53.2.7 update\_scanner()

```
void TcComms::TcPLC::update_scanner ( ) [protected], [virtual]
```

Makes sure we don't have stale values.

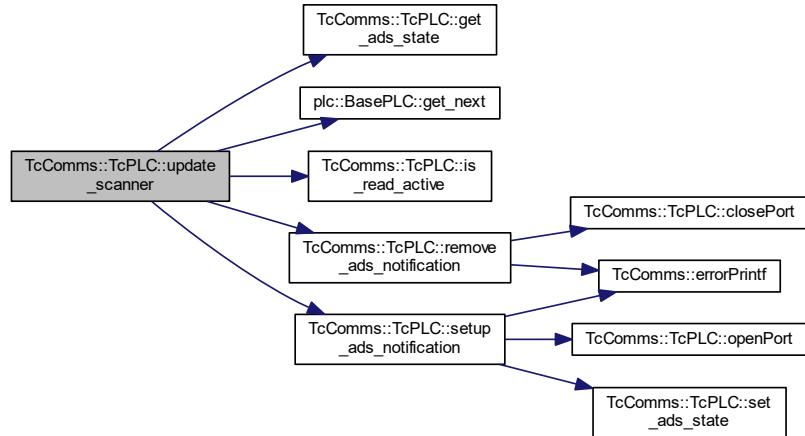
what!?

Reimplemented from [plc::BasePLC](#).

Definition at line 823 of file tcComms.cpp.

References ads\_restart, get\_ads\_state(), plc::BasePLC::get\_next(), is\_read\_active(), plc::BasePLC::mux, plc::BasePLC::name, plc::BasePLC::records, remove\_ads\_notification(), setup\_ads\_notification(), update\_last, and update\_workload.

Here is the call graph for this function:



### 8.53.3 Friends And Related Function Documentation

#### 8.53.3.1 ADSCallback

```
void __stdcall ADSCallback (
    AmsAddr * pAddr,
    AdsNotificationHeader * pNotification,
    unsigned long plcId ) [friend]
```

Notification callback is a friend.

Callback for ADS state change

Definition at line 654 of file tcComms.cpp.

Referenced by setup\_ads\_notification().

### 8.53.4 Member Data Documentation

#### 8.53.4.1 cyclesLeft

```
int TcComms::TcPLC::cyclesLeft [protected]
```

Cycles until EPICS read will be made Counts down from scanRateMultiple, resets at 0

Definition at line 378 of file tcComms.h.

Referenced by read\_scanner().

The documentation for this class was generated from the following files:

- [tcComms.h](#)
- [tcComms.cpp](#)

## 8.54 TcComms::tcProcWrite Class Reference

TwinCAT process write requests.

```
#include <tcComms.h>
```

### Public Member Functions

- `tcProcWrite` (const AmsAddr &a, long amsport, size\_t mrec=1000)  
*Default constructor.*
- `~tcProcWrite` ()  
*Destructor: will porcess the TCat writes.*
- `tcProcWrite` (`tcProcWrite` &&tp) noexcept  
*Move constructor.*
- `void operator()` (`plc::BaseRecord` \*prec)  
*Process on record.*
- `void * read_ptr` (int sz)
- `bool add` (long igrp, long ioffs, long sz)

### Protected Member Functions

- `bool check_alloc` (int extra=0)  
*Checks if we have enough memory allocated.*
- `void tcwrite` ()  
*writes the current header/data to TCat*
- `tcProcWrite & operator=` (`tcProcWrite` &&) noexcept  
*Move operator.*

### Protected Attributes

- `AmsAddr addr`  
*AMS address.*
- `long port`  
*Port to be used to write to TCat.*
- `char * ptr`  
*Pointer to header to be written.*
- `char * data`  
*Pointer to data to be written.*
- `size_t maxrec`  
*Maximum number of individual requests.*
- `size_t size`  
*Size of data.*
- `size_t alloc`  
*Size of allocated header/data array.*
- `size_t count`  
*Current number of individual requests.*
- `std::vector< tcProcWrite > req`

### 8.54.1 Detailed Description

TwinCAT process write requests.

Class for collecting and processing write requests This class iterates through the entire record list on the PLC and collects those records whose data value has a dirty flag set on the plc side. These records are then sent as a group to ADS.

In order to not overload the ADS server, a maximum number of symbols per request is defined, and should not be > 2000.

Definition at line 177 of file tcComms.h.

### 8.54.2 Member Function Documentation

#### 8.54.2.1 add()

```
bool TcComms::tcProcWrite::add (
    long igrp,
    long ioffs,
    long sz )
```

Add header info

##### Parameters

|              |                             |
|--------------|-----------------------------|
| <i>igrp</i>  | iGroup number for tc write  |
| <i>ioffs</i> | iOffset number for tc write |
| <i>sz</i>    | Size of data to be written  |

##### Returns

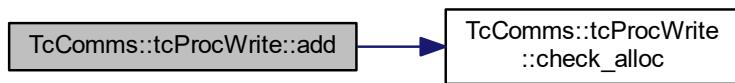
true if successful

Definition at line 245 of file tcComms.cpp.

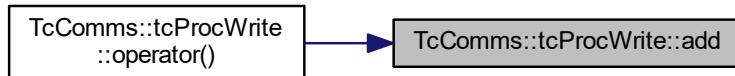
References check\_alloc(), count, maxrec, ptr, and req.

Referenced by operator()().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.54.2.2 read\_ptr()

```
void * TcComms::tcProcWrite::read_ptr (\n    int sz )
```

Get a pointer to read the value in

#### Parameters

|    |                |
|----|----------------|
| sz | Requested size |
|----|----------------|

Definition at line 233 of file tcComms.cpp.

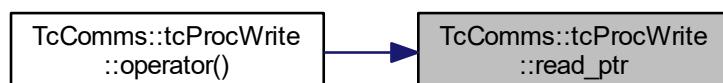
References check\_alloc(), data, and size.

Referenced by operator()().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.54.3 Member Data Documentation

#### 8.54.3.1 req

```
std::vector<tcProcWrite> TcComms::tcProcWrite::req [protected]
```

Queued TCat requests (each would have reached maxrec individual requests)

Definition at line 224 of file tcComms.h.

Referenced by add(), and ~tcProcWrite().

The documentation for this class was generated from the following files:

- [tcComms.h](#)
- [tcComms.cpp](#)

## 8.55 DevTc::tcRegisterToLocShell Class Reference

Register TC commands.

```
#include <drvTc.h>
```

### 8.55.1 Detailed Description

Register TC commands.

Register TC commands to IOC shell This class registers the callback functions for the TC IOC commands

Definition at line 18 of file drvTc.h.

The documentation for this class was generated from the following files:

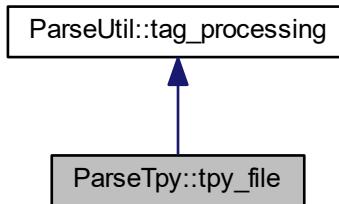
- [drvTc.h](#)
- [drvTc.cpp](#)

## 8.56 ParseTpy::tpy\_file Class Reference

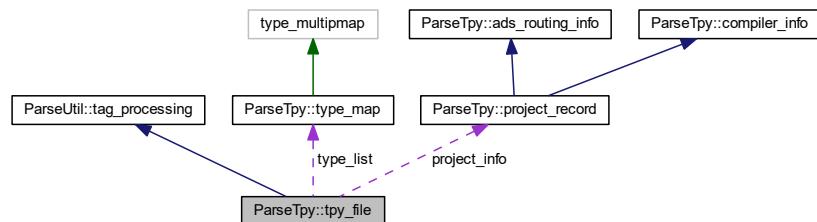
Tpy file parsing.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::tpy\_file:



Collaboration diagram for ParseTpy::tpy\_file:



### Public Member Functions

- [tpy\\_file \(\)](#)  
*Default constructor.*
- [tpy\\_file \(FILE \\*inp\)](#)  
*Constructor.*
- [bool parse \(FILE \\*inp\)](#)  
*Parse a file.*
- [bool parse \(const char \\*p, int len\)](#)  
*Parse a memory region.*
- [const symbol\\_list & get\\_symbols \(\) const](#)  
*Return list of symbols.*
- [const type\\_map & get\\_types \(\) const](#)  
*Return list of types.*
- [const project\\_record & get\\_project\\_info \(\) const](#)

- Return project information.*
- template<class Function>
 

```
int process_symbols (Function &process, const std::stringcase &prefix=std::stringcase()) const
    Process the type tree of a symbol.
```
  - template<class Function>
 

```
int process_type_tree (const symbol_record &symbol, Function &process, const std::stringcase &prefix=std::stringcase()) const
    Process the type tree of a symbol.
```
  - template<class Function>
 

```
int process_type_tree (const type_record &typ, ParseUtil::opc_list defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname=ParseUtil::variable_name(), int level=0) const
    Process the type tree of a type.
```
  - template<class Function>
 

```
int process_type_tree (const std::stringcase &typ, unsigned int id, const ParseUtil::opc_list &defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname=ParseUtil::variable_name(), int level=0) const
    Process the type tree of a type.
```

## Protected Member Functions

- void `parse_finish ()`

*finish up the parsing*
- template<class Function>
 

```
int process_array (const type_record &typ, dimensions dim, const ParseUtil::opc_list &defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname, int level) const
    Process the type tree of a type.
```

## Protected Attributes

- `project_record project_info`

*Project information.*
- `symbol_list sym_list`

*List of symbols.*
- `type_map type_list`

*List of types.*

### 8.56.1 Detailed Description

Tpy file parsing.

This class holds the structure of a tpy file

Definition at line 360 of file ParseTpy.h.

### 8.56.2 Member Function Documentation

### 8.56.2.1 parse\_finish()

```
void ParseTpy::tpy_file::parse_finish ( ) [protected]
```

finish up the parsing

This function is called at the end of parsing. Here we set the TC server name in the OPC variables for each symbol

Definition at line 473 of file ParseTpy.cpp.

References ParseUtil::OPC\_PROP\_PLCNAME.

### 8.56.2.2 process\_array()

```
template<class Function >
int ParseTpy::tpy_file::process_array (
    const type_record & typ,
    dimensions dim,
    const ParseUtil::opc_list & defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
    const ParseUtil::variable_name & varname,
    int level ) const [protected]
```

Process the type tree of a type.

Resolves the type information for an array. Calls the process function for each index with an argument of type process\_arg.

#### Parameters

|                |  |
|----------------|--|
| <i>typ</i>     | Name of type to resolve                                |
| <i>dim</i>     | Dimensions of the array                                |
| <i>defopc</i>  | Default list of OPC parameters                         |
| <i>loc</i>     | Memory location of variable                            |
| <i>process</i> | Function class   |
| <i>varname</i> | Name of variable of the specified type (default is "") |
| <i>level</i>   | Recursive level (stops when reaching 100, default 0)   |

#### Returns

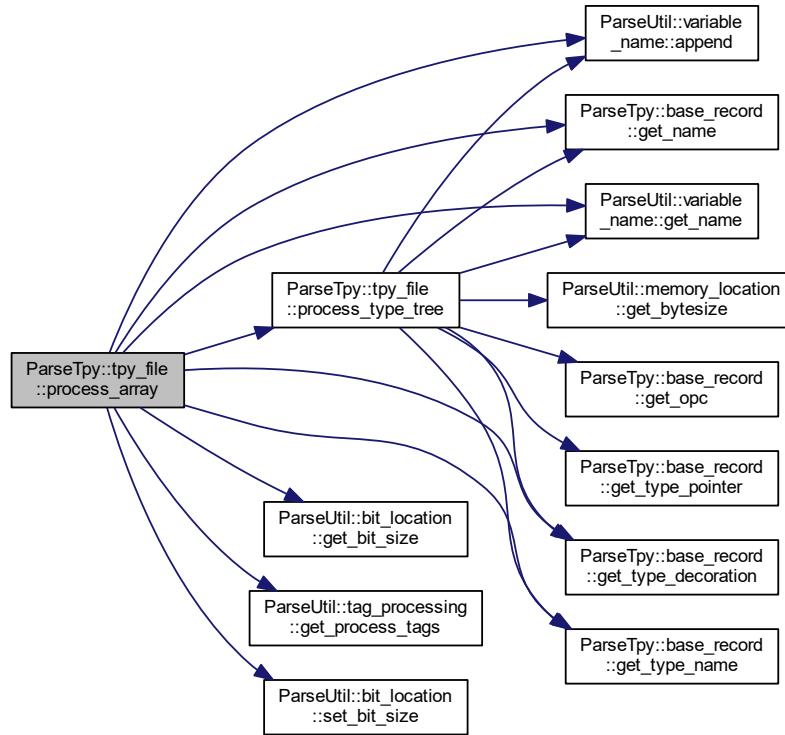
Number of processes variables

Definition at line 213 of file ParseTpyTemplate.h.

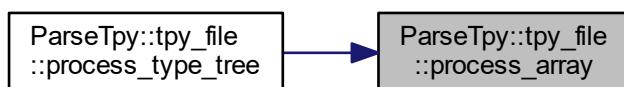
References ParseUtil::variable\_name::append(), ParseUtil::bit\_location::get\_bit\_size(), ParseTpy::base\_record::get\_name(), ParseUtil::variable\_name::get\_name(), ParseUtil::tag\_processing::get\_process\_tags(), ParseTpy::base\_record::get\_type\_decoration(), ParseTpy::base\_record::get\_type\_name(), process\_type\_tree(), and ParseUtil::bit\_location::set\_bit\_size().

Referenced by process\_type\_tree().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.56.2.3 process\_symbols()

```

template<class Function >
int ParseTpy::tpy_file::process_symbols (
    Function & process,
    const std::stringcase & prefix = std::stringcase() ) const
  
```

Process the type tree of a symbol.

Iterates over the symbol list and processes all specified tags.

**Parameters**

|                |   |
|----------------|---|
| <i>process</i> | Function class                              |
| <i>prefix</i>  | Prefix which is added to all variable names |

**Returns**

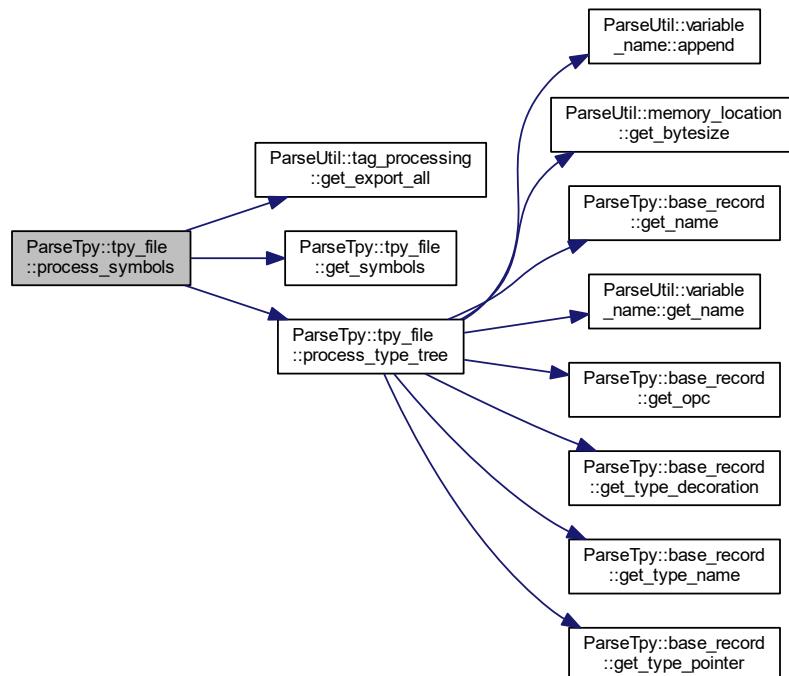
Number of processes variables

Definition at line 9 of file ParseTpyTemplate.h.

References ParseUtil::tag\_processing::get\_export\_all(), get\_symbols(), and process\_type\_tree().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.56.2.4 process\_type\_tree() [1/3]

```
template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const std::string& typ,
    unsigned int id,
    const ParseUtil::opc_list & defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
    const ParseUtil::variable_name & varname = ParseUtil::variable_name(),
    int level = 0 ) const
```

Process the type tree of a type.

Starts with a type and resolves the type information recursively until an atomic type (like STRING) is found. Then, calls the process function with an argument of type process\_arg.

##### Parameters

|                |  |
|----------------|--|
| <i>typ</i>     | Name of type to resolve                                |
| <i>id</i>      | Decoration or unique ID of type                        |
| <i>defopc</i>  | Default list of OPC parameters                         |
| <i>loc</i>     | Memory location of variable                            |
| <i>process</i> | Function class   |
| <i>varname</i> | Name of variable of the specified type (default is "") |
| <i>level</i>   | Recursive level (stops when reaching 100, default 0)   |

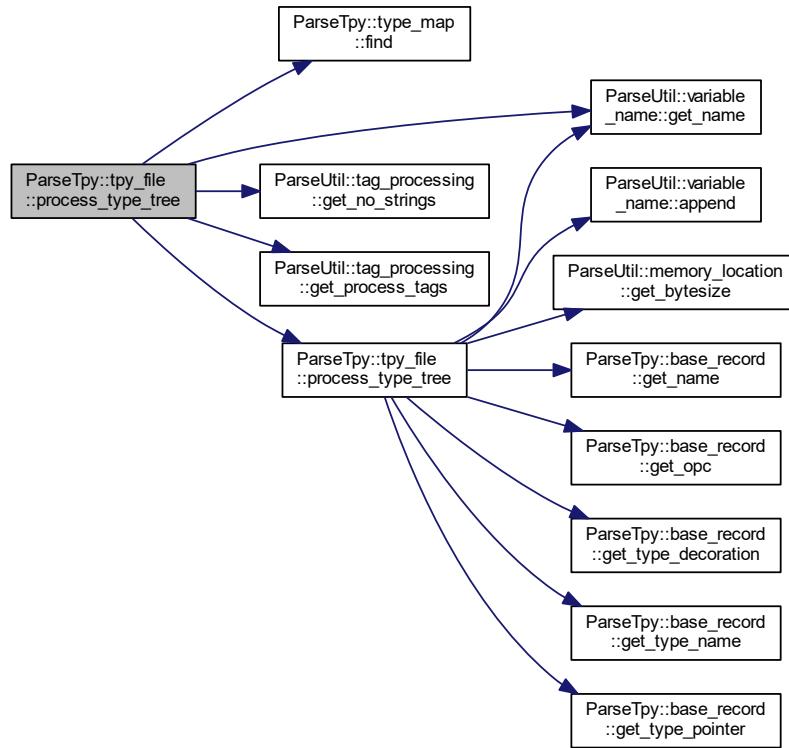
##### Returns

Number of processes variables

Definition at line 160 of file ParseTpyTemplate.h.

References ParseTpy::type\_map::find(), ParseUtil::variable\_name::get\_name(), ParseUtil::tag\_processing::get\_no\_strings(), ParseUtil::tag\_processing::get\_process\_tags(), process\_type\_tree(), and type\_list.

Here is the call graph for this function:



### 8.56.2.5 process\_type\_tree() [2/3]

```

template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const symbol_record & symbol,
    Function & process,
    const std::stringcase & prefix = std::stringcase() ) const
  
```

Process the type tree of a symbol.

Starts with a symbol and resolves the type information recursively until an atomic type (like INT) is found. Then, calls the process function with an argument of type `process_arg`. The function must return true if successful and false otherwise.

#### Parameters

|                      |   |
|----------------------|---|
| <code>symbol</code>  | Symbol to resolve                           |
| <code>process</code> | Function class                              |
| <code>prefix</code>  | Prefix which is added to all variable names |

**Returns**

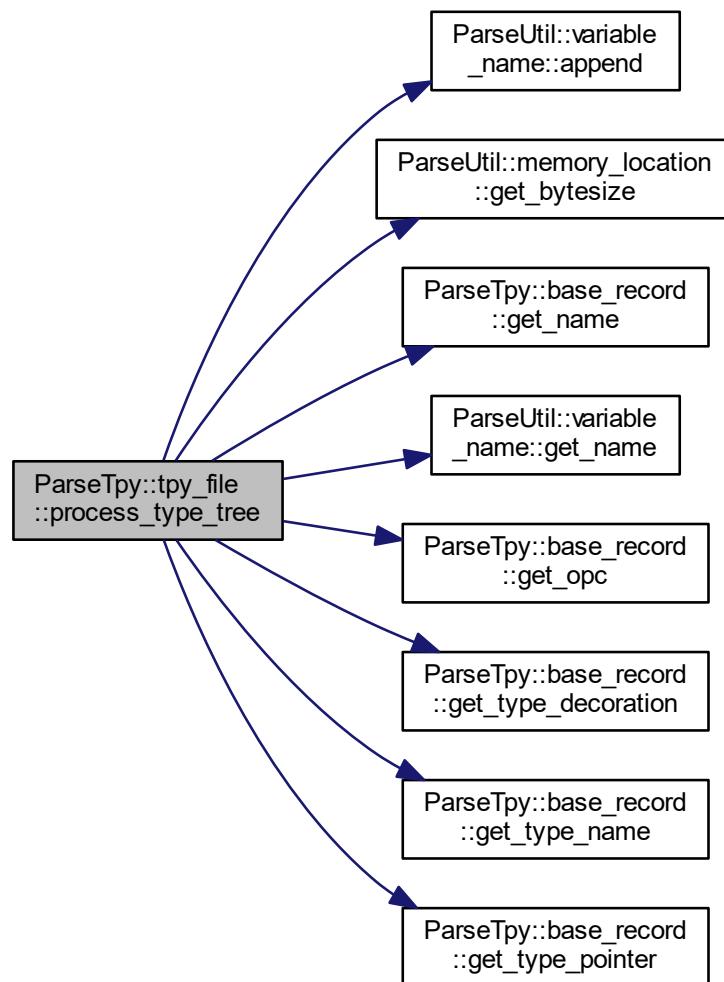
Number of processes variables

Definition at line 25 of file ParseTpyTemplate.h.

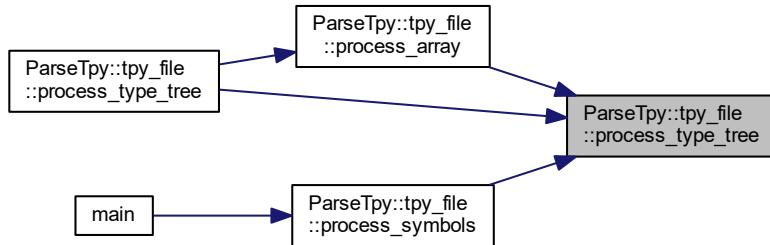
References ParseUtil::variable\_name::append(), ParseUtil::memory\_location::get\_bytesize(), ParseTpy::base\_record::get\_name(), ParseUtil::variable\_name::get\_name(), ParseTpy::base\_record::get\_opc(), ParseTpy::base\_record::get\_type\_decoration(), ParseTpy::base\_record::get\_type\_name(), and ParseTpy::base\_record::get\_type\_pointer().

Referenced by process\_array(), process\_symbols(), and process\_type\_tree().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.56.2.6 process\_type\_tree() [3/3]

```

template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const type_record & typ,
    ParseUtil::opc_list defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
    const ParseUtil::variable_name & varname = ParseUtil::variable_name(),
    int level = 0 ) const
  
```

Process the type tree of a type.

Starts with a type and resolves the type information recursively until an atomic type (like BOOL) is found. Then, calls the process function with an argument of type process\_arg.

#### Parameters

|                |  |
|----------------|--|
| <i>typ</i>     | Type to resolve  |
| <i>defopc</i>  | Default list of OPC parameters                         |
| <i>loc</i>     | Memory location of variable                            |
| <i>process</i> | Function class   |
| <i>varname</i> | Name of variable of the specified type (default is "") |
| <i>level</i>   | Recursive level (stops when reaching 100, default 0)   |

#### Returns

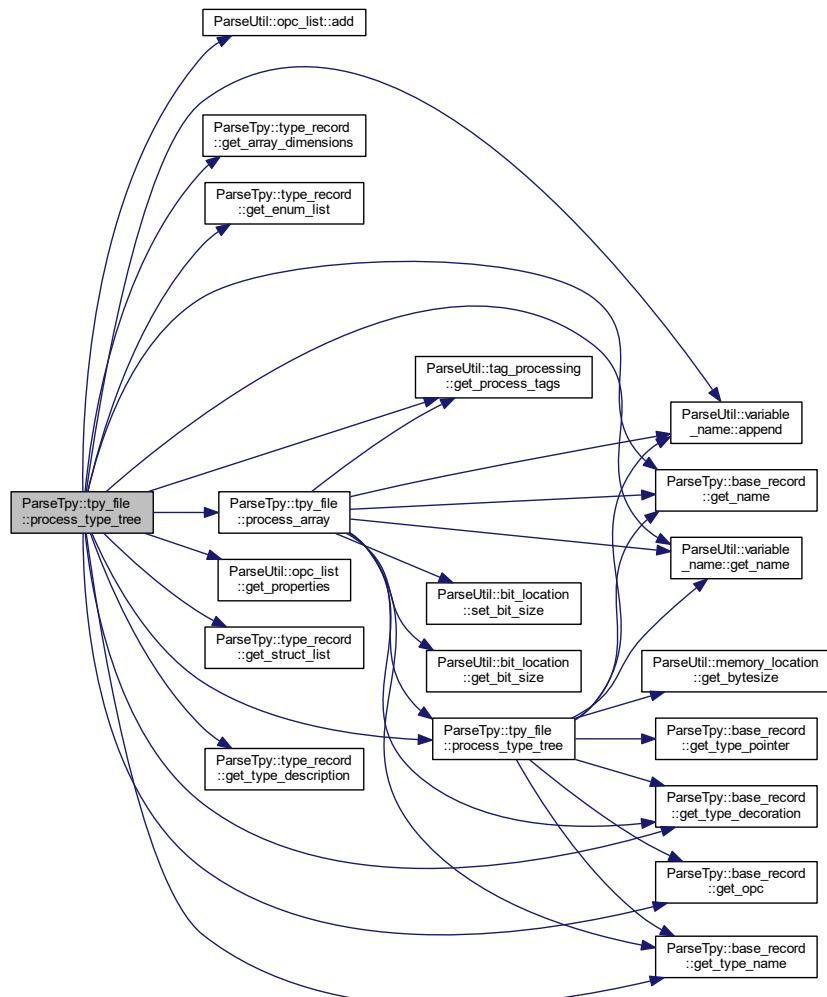
Number of processes variables

Definition at line 55 of file ParseTpyTemplate.h.

References ParseUtil::opc\_list::add(), ParseUtil::variable\_name::append(), ParseTpy::arraytype, ParseTpy::enumtype, ParseTpy::functionblock, ParseTpy::type\_record::get\_array\_dimensions(), ParseTpy::type\_record::

::get\_enum\_list(), ParseTpy::base\_record::get\_name(), ParseUtil::variable\_name::get\_name(), ParseTpy::base\_record::get\_opc(), ParseUtil::tag\_processing::get\_process\_tags(), ParseUtil::opc\_list::get\_properties(), ParseTpy::type\_record::get\_struct\_list(), ParseTpy::base\_record::get\_type\_decoration(), ParseTpy::type\_record::get\_type\_description(), ParseTpy::base\_record::get\_type\_name(), process\_array(), process\_type\_tree(), ParseTpy::simple, and ParseTpy::structtype.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)
- [ParseTpyTemplate.h](#)

## 8.57 DevTc::epics\_record\_traits< RecType >::traits\_type Struct Reference

Epics record type.

```
#include <devTc.h>
```

## Public Attributes

- double **val**

*Value.*

### 8.57.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::epics_record_traits< RecType >::traits_type
```

Epics record type.

Definition at line 290 of file devTc.h.

The documentation for this struct was generated from the following file:

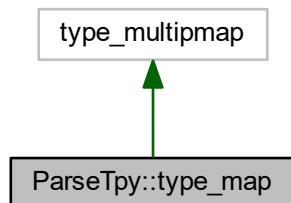
- [devTc.h](#)

## 8.58 ParseTpy::type\_map Class Reference

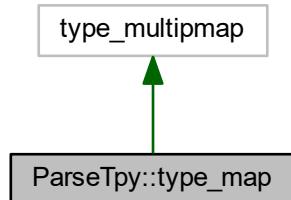
Type dictionary.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::type\_map:



Collaboration diagram for ParseTpy::type\_map:



## Public Member Functions

- [type\\_map \(\)](#)  
*Constructor.*
- void [insert \(value\\_type val\)](#)  
*insert a new element*
- const value\_type::second\_type \* [find \(value\\_type::first\\_type id, const std::string& typn\) const](#)  
*find an element*

### 8.58.1 Detailed Description

Type dictionary.

This is a map of type records, index is type number as defined in tpy

Definition at line 321 of file ParseTpy.h.

The documentation for this class was generated from the following files:

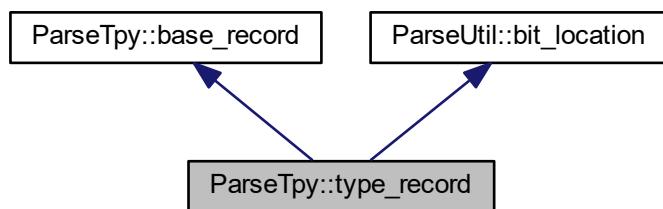
- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

## 8.59 ParseTpy::type\_record Class Reference

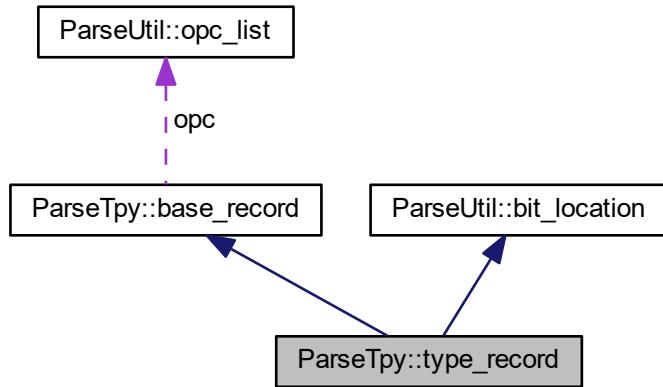
Type record information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::type\_record:



Collaboration diagram for ParseTpy::type\_record:



## Public Member Functions

- `type_record ()`  
*Default constructor.*
- `type_enum get_type_description () const`  
*get the data type*
- `void set_type_description (type_enum desc)`  
*Set decoration (type ID)*
- `unsigned int get_name_decoration () const`  
*Get decoration (type ID)*
- `void set_name_decoration (unsigned int id)`  
*Set decoration (type ID)*
- `const dimensions & get_array_dimensions () const`  
*Get array dimensions.*
- `dimensions & get_array_dimensions ()`  
*Get array dimensions.*
- `const enum_map & get_enum_list () const`  
*Get enumerated list.*
- `enum_map & get_enum_list ()`  
*Get enumerated list.*
- `const item_list & get_struct_list () const`  
*Get structure list.*
- `item_list & get_struct_list ()`  
*Get structure list.*

## Protected Attributes

- `type_enum type_desc`  
*Type description.*
- `unsigned int name_decoration`  
*decoration (type ID) of type name*
- `dimensions array_list`  
*table of dimensions*
- `enum_map enum_list`  
*map of enum id and name*
- `item_list struct_subitems`  
*list of structure elements*

### 8.59.1 Detailed Description

Type record information.

This structure holds a type record

Definition at line 272 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

## 8.60 ParseUtil::variable\_name Class Reference

Variable name.

```
#include <ParseUtil.h>
```

### Public Member Functions

- `variable_name ()`  
*Default constructor.*
- `variable_name (const std::stringcase &n)`  
*Constructor.*
- `variable_name (const char *s)`  
*Constructor.*
- `variable_name (const std::stringcase &n, const std::stringcase &a)`  
*Constructor.*
- `const std::stringcase & get_name () const`  
*Get name.*
- `const std::stringcase & get_alias () const`  
*Get alias.*
- `void set (const std::stringcase &n)`  
*Set name & alias.*
- `void set (const std::stringcase &n, const std::stringcase &a)`  
*Set name & alias.*
- `void append (const std::stringcase &n, const std::stringcase &sep=".")`  
*Append.*
- `void append (const std::stringcase &n, const opc_list &opc, const std::stringcase &sep=".")`  
*Append.*

## Protected Attributes

- `std::stringcase name`  
*variable name*
- `std::stringcase alias`  
*alias name*

### 8.60.1 Detailed Description

Variable name.

This is a class for storing a variable name and an alias

Definition at line 192 of file ParseUtil.h.

The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

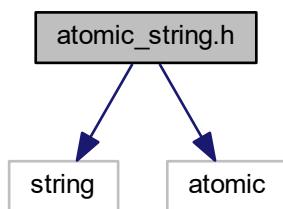


# Chapter 9

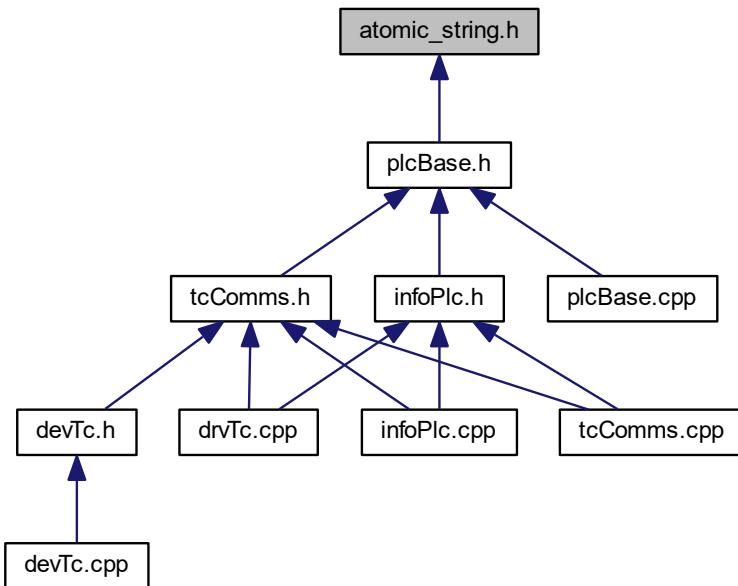
## File Documentation

### 9.1 atomic\_string.h File Reference

```
#include <string>
#include <atomic>
Include dependency graph for atomic_string.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class `std::atomic_string< stringT >`  
*atomic strings*
- class `std::atomic< string >`  
*atomic<string>*
- class `std::atomic< wstring >`  
*atomic<wstring>*

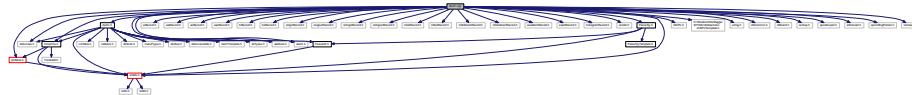
### 9.1.1 Detailed Description

Header which includes a class for implementing an atomic specialization for string.

## 9.2 devTc.cpp File Reference

```
#include "devTc.h"
#include "ParseTpy.h"
#include "InfoPlc.h"
#include "errlog.h"
#include "dbAccess.h"
#include "dbCommon.h"
#include "dbEvent.h"
#include "recSup.h"
```

```
#include "epicsExport.h"
#include "aitConvert.h"
#include "epicsRingPointer.h"
#include <iostream>
Include dependency graph for devTc.cpp:
```



## Namespaces

- [DevTc](#)  
*Namespace for TCat device support.*

## Functions

- [epicsExportAddress \(dset, aival\\_record\\_tc\\_dset\)](#)  
*Record processing entry for ai.*
- [epicsExportAddress \(dset, airval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for raw ai.*
- [epicsExportAddress \(dset, aaival\\_record\\_tc\\_dset\)](#)  
*Record processing entry for aai.*
- [epicsExportAddress \(dset, bival\\_record\\_tc\\_dset\)](#)  
*Record processing entry for bi.*
- [epicsExportAddress \(dset, birval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for raw bi.*
- [epicsExportAddress \(dset, longinval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for longin.*
- [epicsExportAddress \(dset, mbbival\\_record\\_tc\\_dset\)](#)  
*Record processing entry for mbbi.*
- [epicsExportAddress \(dset, mbbirval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for raw mbbi.*
- [epicsExportAddress \(dset, mbbiDirectval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for mbbiDirect.*
- [epicsExportAddress \(dset, mbbiDirectrval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for raw mbbiDirect.*
- [epicsExportAddress \(dset, stringinval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for stringin.*
- [epicsExportAddress \(dset, waveformval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for waveform.*
- [epicsExportAddress \(dset, aoaval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for ao.*
- [epicsExportAddress \(dset, aorval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for raw ao.*
- [epicsExportAddress \(dset, aaoval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for aao.*
- [epicsExportAddress \(dset, boaval\\_record\\_tc\\_dset\)](#)  
*Record processing entry for bo.*

- **epicsExportAddress** (dset, borval\_record\_tc\_dset)  
*Record processing entry for raw bo.*
- **epicsExportAddress** (dset, longoutval\_record\_tc\_dset)  
*Record processing entry for longout.*
- **epicsExportAddress** (dset, mbboval\_record\_tc\_dset)  
*Record processing entry for mbbo.*
- **epicsExportAddress** (dset, mbborval\_record\_tc\_dset)  
*Record processing entry for raw mbbo.*
- **epicsExportAddress** (dset, mbboDirectval\_record\_tc\_dset)  
*Record processing entry for mbboDirect.*
- **epicsExportAddress** (dset, mbboDirectrval\_record\_tc\_dset)  
*Record processing entry for raw mbboDirect.*
- **epicsExportAddress** (dset, stringoutval\_record\_tc\_dset)  
*Record processing entry for stringout.*

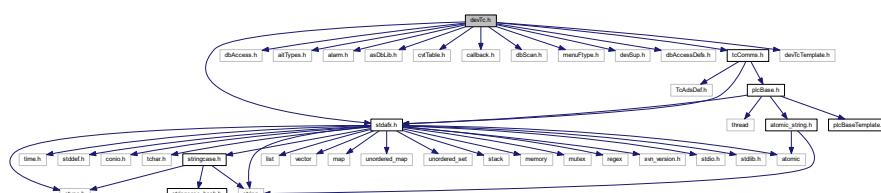
### 9.2.1 Detailed Description

Methods for TwinCAT/ADS device support.

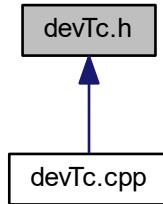
## 9.3 devTc.h File Reference

```
#include "stdafx.h"
#include "dbAccess.h"
#include "aitTypes.h"
#include "alarm.h"
#include "asDbLib.h"
#include "cvtTable.h"
#include "callback.h"
#include "dbScan.h"
#include "menuFtype.h"
#include "devSup.h"
#include "dbAccessDefs.h"
#include "tcComms.h"
#include "devTcTemplate.h"

Include dependency graph for devTc.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [DevTc::register\\_devsup](#)  
*Device support registration.*
- class [DevTc::EpicsInterface](#)  
*Epics interface class.*
- struct [DevTc::epics\\_record\\_traits< RecType >](#)  
*Epics record traits.*
- struct [DevTc::epics\\_record\\_traits< RecType >::traits\\_type](#)  
*Epics record type.*
- struct [DevTc::devTcDeflo< RecType >](#)  
*Device support record.*
- struct [DevTc::devTcDefIn< RecType >](#)  
*Device support input record.*
- struct [DevTc::devTcDefOut< RecType >](#)  
*device support output record.*
- struct [DevTc::devTcDefWaveformIn< RecType >](#)  
*device support waveform record.*

## Namespaces

- [DevTc](#)  
*Namespace for TCat device support.*

## Enumerations

- enum [DevTc::epics\\_record\\_enum](#) {  
    DevTc::aaival = 0, DevTc::aoval, DevTc::aival, DevTc::aoval,  
    DevTc::bival, DevTc::boval, DevTc::eventval, DevTc::histogramval,  
    DevTc::longinval, DevTc::longoutval, DevTc::mbbival, DevTc::mbboval,  
    DevTc::mbbiDirectval, DevTc::mbboDirectval, DevTc::stringinval, DevTc::stringoutval,  
    DevTc::waveformval, DevTc::airval, DevTc::aorval, DevTc::birval,  
    DevTc::borval, DevTc::mbbiDirectval, DevTc::mbboDirectval, DevTc::mbbirval,  
    DevTc::mbborval, DevTc::epics\_record\_end, DevTc::invalidval = -1 }  
*Epics record type enum.*

## Functions

- const std::regex [DevTc::tc\\_regex](#) ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.|.?)+:([8[0-9][0-9]))|/([\\d{1,9}])/([\\d{1,9}]):([\\d{1,9}])")  
*Callback for output record.*
- const std::regex [DevTc::info\\_regex](#) ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.|.?)+:([8[0-9][0-9]))|/([info])/([A-Za-z0-9\_]+)")  
*Regex for identifying info records.*

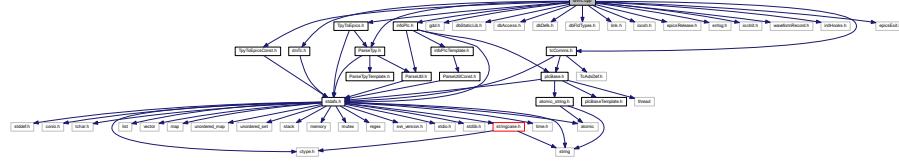
### 9.3.1 Detailed Description

Header which includes classes for TwinCAT/ADS device support.

## 9.4 drvTc.cpp File Reference

```
#include "drvTc.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
#include "TpyToEpicsConst.h"
#include "infoPlc.h"
#include "gdd.h"
#include "dbStaticLib.h"
#include "dbAccess.h"
#include "dbDefs.h"
#include "dbFldTypes.h"
#include "link.h"
#include "iocsh.h"
#include "epicsRelease.h"
#include "errlog.h"
#include "iocInit.h"
#include "waveformRecord.h"
#include "initHooks.h"
#include "tcComms.h"
#include "epicsExit.h"
```

Include dependency graph for drvTc.cpp:



## Classes

- class [DevTc::epics\\_tc\\_db\\_processing](#)  
*EPICS/TCat db processing.*

## Namespaces

- **DevTc**

*Namespace for TCat device support.*

## Typedefs

- `typedef std::tuple< std::stringcase, std::stringcase, epics_list_processing *, bool > DevTc::filename_rule_list_tuple`  
*Tuple for filnemae, rule and list processing.*
- `typedef std::vector< filename_rule_list_tuple > DevTc::tc_listing_def`  
*List of tuples for filnemae, rule and list processing.*
- `typedef std::tuple< std::stringcase, std::stringcase, epics_macrofiles_processing *, const char * > DevTc::dirname_arg_macro_tuple`  
*Tuple for directory name, argument and macro list processing.*
- `typedef std::vector< dirname_arg_macro_tuple > DevTc::tc_macro_def`  
*List of tuples for directory name, argument and macro list processing.*

## Functions

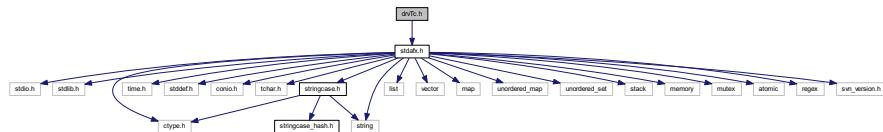
- `void DevTc::tcLoadRecords (const iocshArgBuf *args)`  
*Load TwinCAT records.*
- `void DevTc::tcSetScanRate (const iocshArgBuf *args)`  
*Set the scan rate.*
- `void DevTc::tcList (const iocshArgBuf *args)`  
*Generate channel lists.*
- `void DevTc::tcMacro (const iocshArgBuf *args)`  
*Generated macro files.*
- `void DevTc::tcAlias (const iocshArgBuf *args)`  
*Define alias and replacement rules.*
- `void DevTc::tcInfoPrefix (const iocshArgBuf *args)`  
*Sets the info prefix.*
- `void DevTc::tcPrintVals (const iocshArgBuf *args)`  
*Print all values.*
- `void DevTc::tcPrintVal (const iocshArgBuf *args)`  
*Print value.*

### 9.4.1 Detailed Description

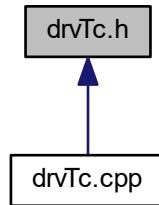
This contains functions for driver support for EPICS. These routines are used during the IOC initialization, and allow the IOC to initialize the TwinCAT interface during EPICS initialization.

## 9.5 drvTc.h File Reference

```
#include "stdafx.h"
Include dependency graph for drvTc.h:
```



This graph shows which files directly or indirectly include this file:



### Classes

- class [DevTc::tcRegisterToLocShell](#)

*Register TC commands.*

### Namespaces

- [DevTc](#)

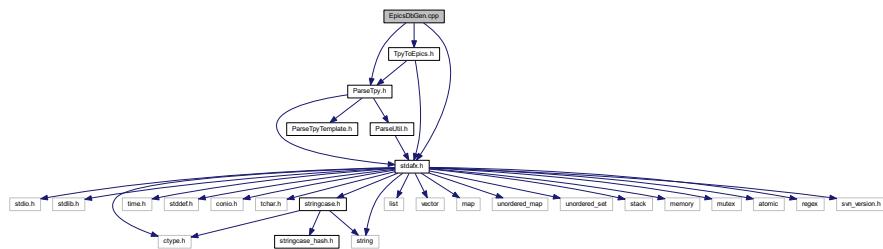
*Namespace for TCat device support.*

### 9.5.1 Detailed Description

Header which includes classes for accessing TwinCAT/ADS from EPICS.

## 9.6 EpicsDbGen.cpp File Reference

```
#include "stdaafx.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
Include dependency graph for EpicsDbGen.cpp:
```



## Functions

- int [main](#) (int argc, char \*argv[])

### 9.6.1 Detailed Description

Source for the main program that generates an EPICs .db file

### 9.6.2 Function Documentation

#### 9.6.2.1 [main\(\)](#)

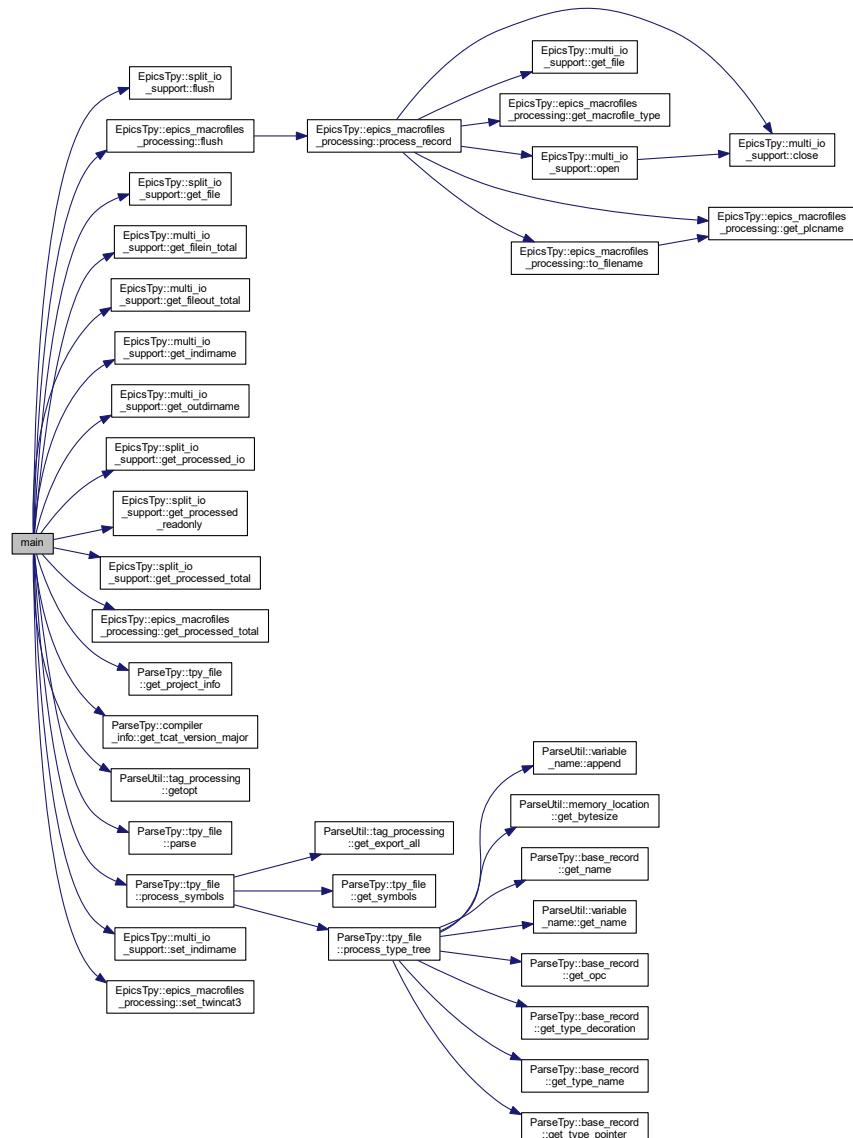
```
int main (
    int argc,
    char * argv[ ] )
```

Main program

Definition at line 17 of file EpicsDbGen.cpp.

References `EpicsTpy::split_io_support::flush()`, `EpicsTpy::epics_macrofiles_processing::flush()`, `EpicsTpy::split_io_support::get_file()`, `EpicsTpy::multi_io_support::get_filein_total()`, `EpicsTpy::multi_io_support::get_fileout_total()`, `EpicsTpy::multi_io_support::get_indirname()`, `EpicsTpy::multi_io_support::get_outdirname()`, `EpicsTpy::split_io_support::get_processed_io()`, `EpicsTpy::split_io_support::get_processed_READONLY()`, `EpicsTpy::split_io_support::get_processed_total()`, `EpicsTpy::epics_macrofiles_processing::get_processed_total()`, `ParseTpy::tpy_file::get_project_info()`, `ParseTpy::compiler_info::get_tcat_version_major()`, `ParseUtil::tag_processing::getopt()`, `ParseTpy::tpy_file::parse()`, `ParseTpy::tpy_file::process_symbols()`, `EpicsTpy::multi_io_support::set_indirname()`, and `EpicsTpy::epics_macrofiles_processing::set_twincat3()`.

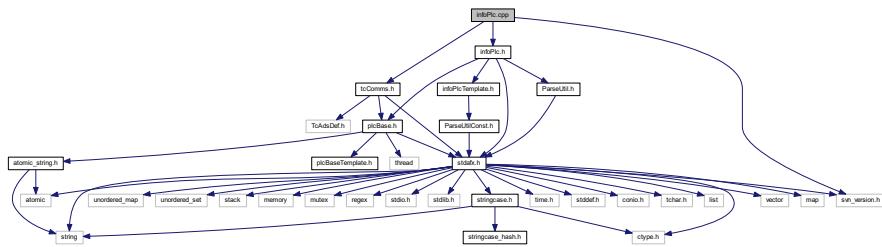
Here is the call graph for this function:



## 9.7 infoPlc.cpp File Reference

```
#include "infoPlc.h"
#include "tcComms.h"
#include "svn_version.h"
```

Include dependency graph for infoPlc.cpp:



## Namespaces

- **InfoPlc**

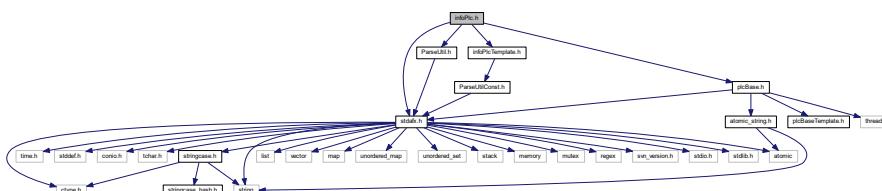
*Namespace for Info communication.*

### 9.7.1 Detailed Description

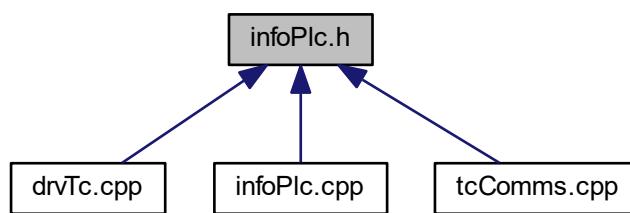
Defines methods for the info PLC.

## 9.8 infoPlc.h File Reference

```
#include "stdafx.h"
#include "ParseUtil.h"
#include "plcBase.h"
#include "infoPlcTemplate.h"
Include dependency graph for infoPlc.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [InfoPlc::InfoInterface](#)  
*Info interface.*
- class [InfoPlc::process\\_arg\\_info](#)  
*Arguments for processing.*

## Namespaces

- [InfoPlc](#)  
*Namespace for Info communication.*

## TypeDefs

- typedef bool([InfoInterface](#)::\* [InfoPlc::info\\_update\\_method](#)) ()  
*Pointer to info update method.*
- typedef std::tuple< [ParseUtil::variable\\_name](#), [ParseUtil::process\\_type\\_enum](#), [ParseUtil::opc\\_list](#), std::stringcase, bool, update\_enum, info\_update\_method > [InfoPlc::info\\_dbrecord\\_type](#)
- typedef std::vector< [info\\_dbrecord\\_type](#) > [InfoPlc::info\\_dbrecord\\_list](#)  
*List type of db info tuples.*

## Enumerations

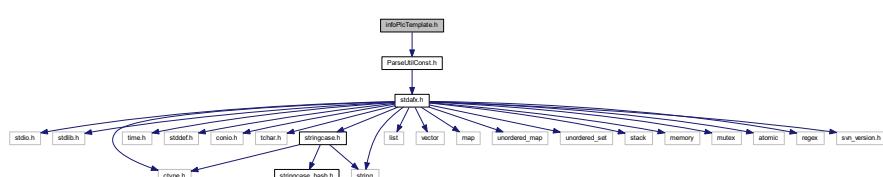
- enum [InfoPlc::update\\_enum](#) { [InfoPlc::update\\_enum::forever](#), [InfoPlc::update\\_enum::once](#), [InfoPlc::update\\_enum::done](#) }
- Update frequency type.*

### 9.8.1 Detailed Description

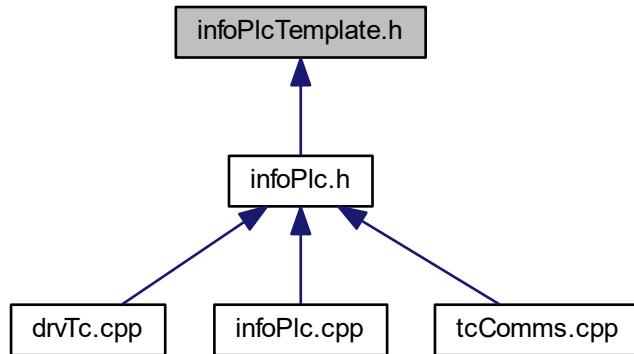
Header which includes classes for the info PLC.

## 9.9 infoPlcTemplate.h File Reference

```
#include "ParseUtilConst.h"
Include dependency graph for infoPlcTemplate.h:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- [InfoPlc](#)  
*Namespace for Info communication.*

### 9.9.1 Detailed Description

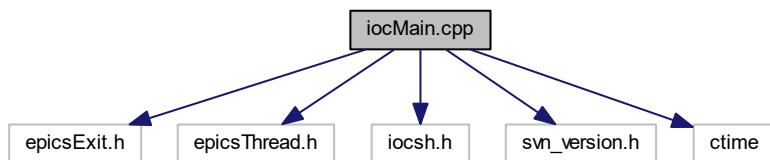
Header which includes classes for the info PLC.

## 9.10 iocMain.cpp File Reference

```

#include "epicsExit.h"
#include "epicsThread.h"
#include "iocsh.h"
#include "svn_version.h"
#include <ctime>
  
```

Include dependency graph for `iocMain.cpp`:



## Functions

- `__declspec (dllexport) void stopTc(void)`  
*DLL import for stopTc.*
- int `main` (int argc, char \*argv[ ])  
`tcloc`

### 9.10.1 Detailed Description

The main program for the TwinCAT IOC.

### 9.10.2 Function Documentation

#### 9.10.2.1 main()

```
int main (
    int argc,
    char * argv[ ] )
```

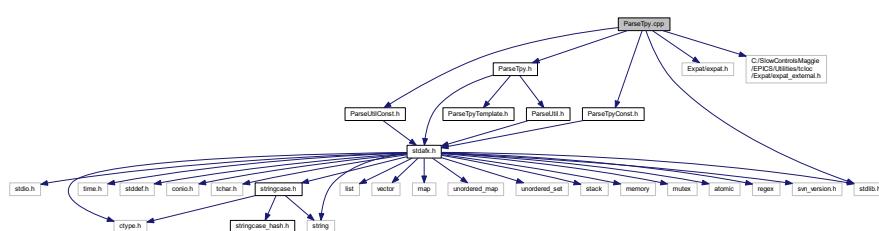
`tcloc`

Main program for `tcloc`

Definition at line 27 of file `iocMain.cpp`.

## 9.11 ParseTpy.cpp File Reference

```
#include "ParseTpy.h"
#include "ParseUtilConst.h"
#include "ParseTpyConst.h"
#include "Expat/expat.h"
Include dependency graph for ParseTpy.cpp:
```



## Classes

- class `ParseTpy::parserinfo_type`  
*Parser information.*

## Namespaces

- `ParseTpy`

*Namespace for parsing.*

## Macros

- `#define XML_STATIC`  
*Static linking.*
- `#define XML_FMT_INT_MOD "I"`  
*Int format.*

## Functions

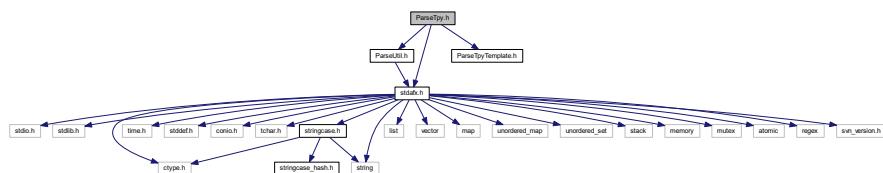
- `bool ParseTpy::compareNamesWoNamespace (const std::stringcase &p1, const std::stringcase &p2)`
- `bool ParseTpy::get_decoration (const char **atts, unsigned int &decoration)`
- `bool ParseTpy::get_pointer (const char **atts)`

### 9.11.1 Detailed Description

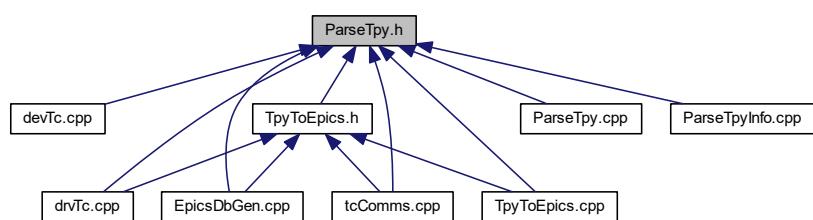
Methods to parse TwinCAT tpy file.

## 9.12 ParseTpy.h File Reference

```
#include "stdafx.h"
#include "ParseUtil.h"
#include "ParseTpyTemplate.h"
Include dependency graph for ParseTpy.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [ParseTpy::ads\\_routing\\_info](#)  
*ADS routing information.*
- class [ParseTpy::compiler\\_info](#)  
*Compiler information.*
- class [ParseTpy::project\\_record](#)  
*Project information.*
- class [ParseTpy::base\\_record](#)  
*Base record definition.*
- class [ParseTpy::item\\_record](#)  
*item record*
- class [ParseTpy::type\\_record](#)  
*Type record information.*
- class [ParseTpy::type\\_map](#)  
*Type dictionary.*
- class [ParseTpy::symbol\\_record](#)  
*Symbol record.*
- class [ParseTpy::tpy\\_file](#)  
*Tpy file parsing.*

## Namespaces

- [ParseTpy](#)  
*Namespace for parsing.*

## Typedefs

- `typedef std::pair< int, int > ParseTpy::dimension`
- `typedef std::list< dimension > ParseTpy::dimensions`
- `typedef std::map< int, std::stringcase > ParseTpy::enum_map`
- `typedef std::pair< int, std::stringcase > ParseTpy::enum_pair`
- `typedef std::list< item_record > ParseTpy::item_list`
- `typedef std::multimap< unsigned int, type_record > ParseTpy::type_multipmap`
- `typedef std::list< symbol_record > ParseTpy::symbol_list`

## Enumerations

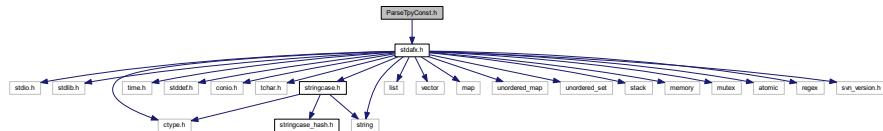
- `enum ParseTpy::type_enum {  
 ParseTpy::type_enum::unknown, ParseTpy::type_enum::simple, ParseTpy::type_enum::arraytype, ParseTpy::type_enum::enumtype,  
 ParseTpy::type_enum::structtype, ParseTpy::type_enum::functionblock }`  
*Type enum.*

### 9.12.1 Detailed Description

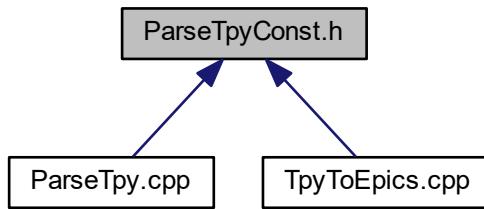
Header which includes classes to parse a TwinCAT tpy file. It includes "ParseUtil.h" and "ParseTpyTemplate.h"

## 9.13 ParseTpyConst.h File Reference

```
#include "stdafx.h"
Include dependency graph for ParseTpyConst.h:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- [ParseUtil](#)  
*Namespace for parsing utilities.*
- [ParseTpy](#)  
*Namespace for parsing.*

## Variables

- `const char *const ParseTpy::xmlPlcProjectInfo = "PlcProjectInfo"`  
*PLC project info.*
- `const char *const ParseTpy::xmlProjectInfo = "ProjectInfo"`  
*Project info.*
- `const char *const ParseTpy::xmlRoutingInfo = "RoutingInfo"`  
*Routing info.*
- `const char *const ParseTpy::xmlCompilerInfo = "CompilerInfo"`  
*Compiler info.*
- `const char *const ParseTpy::xmlAdsInfo = "AdsInfo"`  
*ADS info.*
- `const char *const ParseTpy::xmlDataTypes = "DataTypes"`  
*Data types.*
- `const char *const ParseTpy::xmlDataType = "DataType"`

- const char \*const ParseTpy::xmlSymbols = "Symbols"  
*Symbols.*
- const char \*const ParseTpy::xmlSymbol = "Symbol"  
*Symbol.*
- const char \*const ParseTpy::xmlProperties = "Properties"  
*Properties.*
- const char \*const ParseTpy::xmlProperty = "Property"  
*Property.*
- const char \*const ParseTpy::xmlCompilerVersion = "CompilerVersion"  
*Compiler version.*
- const char \*const ParseTpy::xmlTwinCATVersion = "TwinCATVersion"  
*TwinCAT version.*
- const char \*const ParseTpy::xmlCpuFamily = "CpuFamily"  
*CPU family.*
- const char \*const ParseTpy::xmlNetId = "NetId"  
*Net ID.*
- const char \*const ParseTpy::xmlPort = "Port"  
*Port.*
- const char \*const ParseTpy::xmlTargetName = "TargetName"  
*Target name.*
- const char \*const ParseTpy::xmlName = "Name"  
*Name.*
- const char \*const ParseTpy::xmlType = "Type"  
*Type.*
- const char \*const ParseTpy::xmlAttrDecoration = "Decoration"  
*Decoration.*
- const char \*const ParseTpy::xmlAttrPointer = "Pointer"  
*Pointer.*
- const char \*const ParseTpy::xmlIGroup = "IGroup"  
*I Group.*
- const char \*const ParseTpy::xmlIOffset = "IOffset"  
*I Offset.*
- const char \*const ParseTpy::xmlBitSize = "BitSize"  
*Bit size.*
- const char \*const ParseTpy::xmlBitOffs = "BitOffs"  
*Bit Offset.*
- const char \*const ParseTpy::xmlArrayInfo = "ArrayInfo"  
*Array info.*
- const char \*const ParseTpy::xmlArrayLBound = "LBound"  
*Lower bound.*
- const char \*const ParseTpy::xmlArrayElements = "Elements"  
*Elements.*
- const char \*const ParseTpy::xmlSubItem = "SubItem"  
*Sub item.*
- const char \*const ParseTpy::xmlFbInfo = "FbInfo"  
*Fb info.*
- const char \*const ParseTpy::xmlEnumInfo = "EnumInfo"  
*Enum info.*
- const char \*const ParseTpy::xmlEnumText = "Text"  
*Text.*

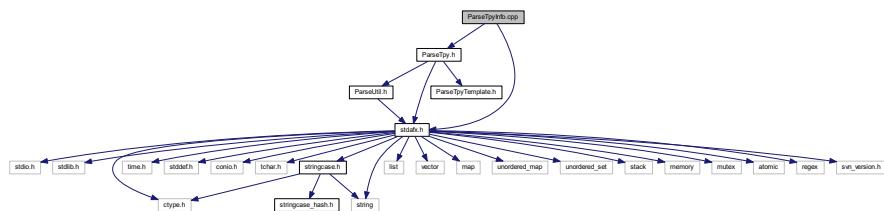
- const char \*const ParseTpy::xmlEnumEnum = "Enum"  
*Enum.*
- const char \*const ParseTpy::xmlEnumComment = "Comment"  
*Comment.*
- const char \*const ParseTpy::xmlValue = "Value"  
*Value.*
- const char \*const ParseTpy::xmlDesc = "Desc"  
*Description.*
- const char \*const ParseTpy::opcExport = "opc"  
*OPC.*
- const char \*const ParseTpy::opcProp = "opc\_prop"  
*OPC property.*
- const char \*const ParseTpy::opcBracket = "["  
*OPC bracket.*

### 9.13.1 Detailed Description

Header which includes constants needed to parse a TwinCAT tpy file.

## 9.14 ParseTpyInfo.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
Include dependency graph for ParseTpyInfo.cpp:
```



## Classes

- class **syminfo\_processing**  
*Symbol processing.*

## Functions

- int **main** (int argc, char \*argv[ ])   
*tpyinfo*

### 9.14.1 Detailed Description

Source file for tpy parsing methods.

## 9.14.2 Function Documentation

### 9.14.2.1 main()

```
int main (
    int argc,
    char * argv[] )
```

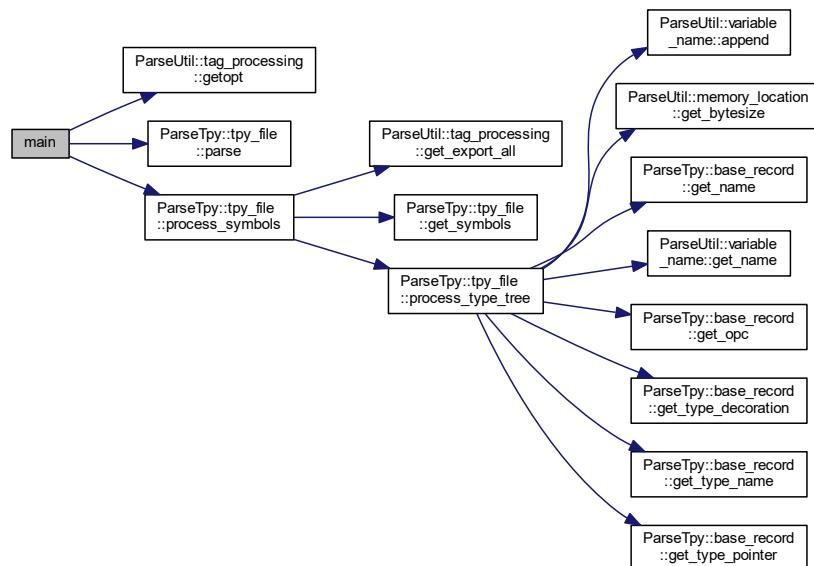
tpyinfo

Main program for tpyinfo

Definition at line 93 of file ParseTpInfo.cpp.

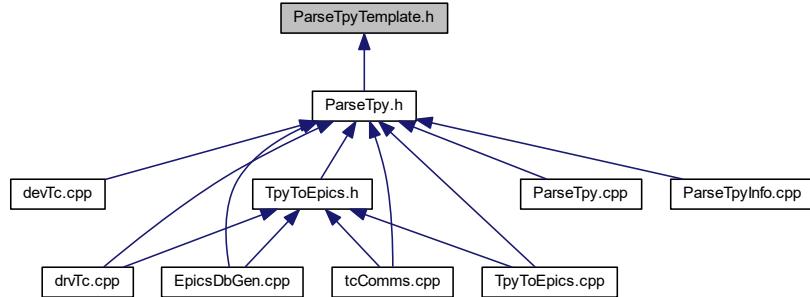
References `ParseUtil::tag_processing::getopt()`, `ParseTp::tpy_file::parse()`, and `ParseTp::tpy_file::process_symbols()`.

Here is the call graph for this function:



## 9.15 ParseTpyTemplate.h File Reference

This graph shows which files directly or indirectly include this file:



### Namespaces

- [ParseTpy](#)

*Namespace for parsing.*

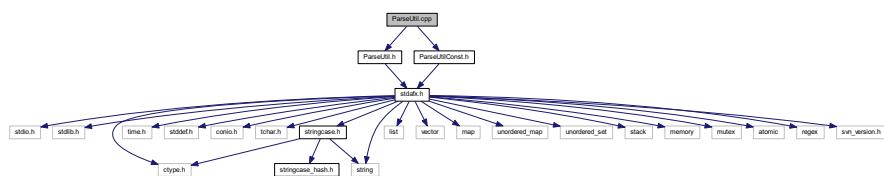
#### 9.15.1 Detailed Description

Templates for functions for parsing tpy file.

## 9.16 ParseUtil.cpp File Reference

```
#include "ParseUtil.h"
#include "ParseUtilConst.h"
```

Include dependency graph for ParseUtil.cpp:



### Namespaces

- [ParseUtil](#)

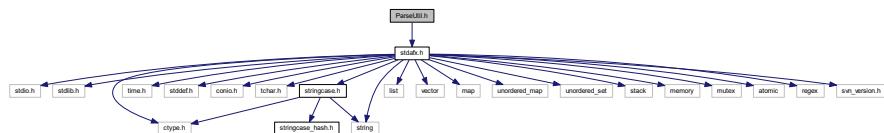
*Namespace for parsing utilities.*

### 9.16.1 Detailed Description

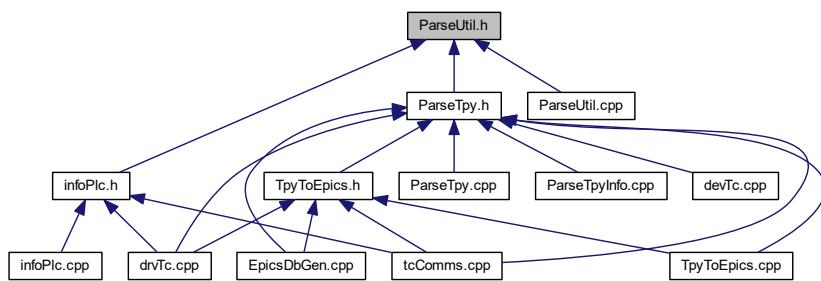
Utility methods to parsing.

## 9.17 ParseUtil.h File Reference

```
#include "stdafx.h"
Include dependency graph for ParseUtil.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [ParseUtil::replacement\\_rules](#)  
*Replacement rules.*
- class [ParseUtil::optarg](#)  
*Optional arguments.*
- class [ParseUtil::opc\\_list](#)  
*OPC list.*
- class [ParseUtil::variable\\_name](#)  
*Variable name.*
- class [ParseUtil::bit\\_location](#)  
*Bit location.*
- class [ParseUtil::memory\\_location](#)  
*Memory location.*
- class [ParseUtil::process\\_arg](#)  
*Arguments for processing.*
- class [ParseUtil::process\\_arg\\_tc](#)  
*Arguments for processing.*
- class [ParseUtil::tag\\_processing](#)  
*Tag processing selection.*

## Namespaces

- `ParseUtil`

*Namespace for parsing utilities.*

## Typedefs

- `typedef std::map< std::stringcase, std::stringcase > ParseUtil::replacement_table`
- `typedef std::map< int, std::stringcase > ParseUtil::property_map`
- `typedef std::pair< int, std::stringcase > ParseUtil::property_el`

## Enumerations

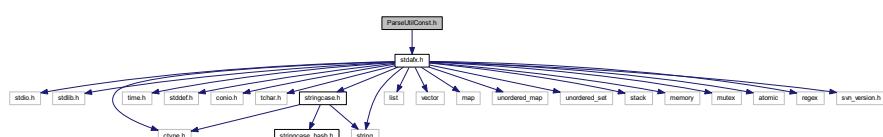
- `enum ParseUtil::opc_enum { ParseUtil::opc_enum::no_change, ParseUtil::opc_enum::publish, ParseUtil::opc_enum::silent }`  
*OPC state enum.*
- `enum ParseUtil::process_type_enum { ParseUtil::process_type_enum::pt_invalid, ParseUtil::process_type_enum::pt_int, ParseUtil::process_type_enum::pt_real, ParseUtil::process_type_enum::pt_bool, ParseUtil::process_type_enum::pt_string, ParseUtil::process_type_enum::pt_enum, ParseUtil::process_type_enum::pt_binary }`  
*Process type.*
- `enum ParseUtil::process_tag_enum { ParseUtil::process_tag_enum::all, ParseUtil::process_tag_enum::atomic, ParseUtil::process_tag_enum::structured }`  
*Tag preoicing enum.*

### 9.17.1 Detailed Description

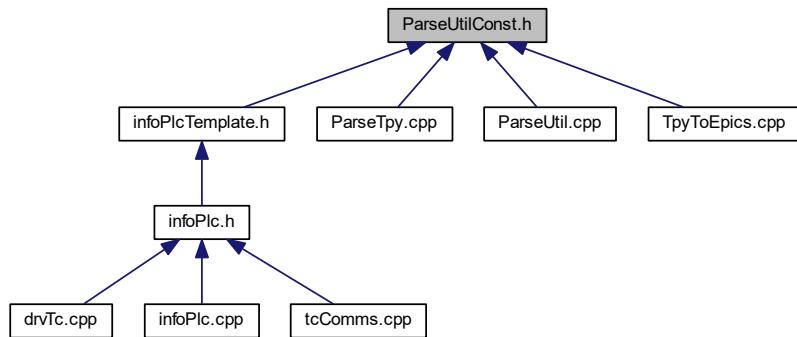
Header which includes utility classes for parsing. It includes "ParseUtilTemplate.h"

## 9.18 ParseUtilConst.h File Reference

```
#include "stdafx.h"
Include dependency graph for ParseUtilConst.h:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- [ParseTpy](#)  
*Namespace for parsing.*
- [ParseUtil](#)  
*Namespace for parsing utilities.*

## Variables

- const int [ParseUtil::OPC\\_PROP\\_CDT](#) = 1
- const int [ParseUtil::OPC\\_PROP\\_VALUE](#) = 2
- const int [ParseUtil::OPC\\_PROP\\_QUALITY](#) = 3
- const int [ParseUtil::OPC\\_PROP\\_TIME](#) = 4
- const int [ParseUtil::OPC\\_PROP\\_RIGHTS](#) = 5
- const int [ParseUtil::OPC\\_PROP\\_SCANRATE](#) = 6
- const int [ParseUtil::OPC\\_PROP\\_UNIT](#) = 100
- const int [ParseUtil::OPC\\_PROP\\_DESC](#) = 101
- const int [ParseUtil::OPC\\_PROP\\_HIEU](#) = 102
- const int [ParseUtil::OPC\\_PROP\\_LOEU](#) = 103
- const int [ParseUtil::OPC\\_PROP\\_HIRANGE](#) = 104
- const int [ParseUtil::OPC\\_PROP\\_LORANGE](#) = 105
- const int [ParseUtil::OPC\\_PROP\\_CLOSE](#) = 106
- const int [ParseUtil::OPC\\_PROP\\_OPEN](#) = 107
- const int [ParseUtil::OPC\\_PROP\\_TIMEZONE](#) = 108
- const int [ParseUtil::OPC\\_PROP\\_FGC](#) = 201
- const int [ParseUtil::OPC\\_PROP\\_BGC](#) = 202
- const int [ParseUtil::OPC\\_PROP\\_BLINK](#) = 203
- const int [ParseUtil::OPC\\_PROP\\_BMP](#) = 204
- const int [ParseUtil::OPC\\_PROP SND](#) = 205
- const int [ParseUtil::OPC\\_PROP\\_HTML](#) = 206
- const int [ParseUtil::OPC\\_PROP\\_AVI](#) = 207
- const int [ParseUtil::OPC\\_PROP\\_ALMSTAT](#) = 300
- const int [ParseUtil::OPC\\_PROP\\_ALMHELP](#) = 301
- const int [ParseUtil::OPC\\_PROP\\_ALMAREAS](#) = 302

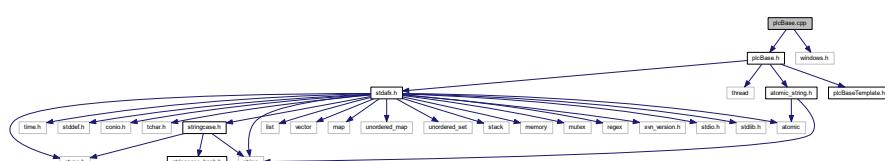
- const int ParseUtil::OPC\_PROP\_ALMPRIMARYAREA = 303
- const int ParseUtil::OPC\_PROP\_ALMCONDITION = 304
- const int ParseUtil::OPC\_PROP\_ALMLIMIT = 305
- const int ParseUtil::OPC\_PROP\_ALMDB = 306
- const int ParseUtil::OPC\_PROP\_ALMH = 307
- const int ParseUtil::OPC\_PROP\_ALMH = 308
- const int ParseUtil::OPC\_PROP\_ALML = 309
- const int ParseUtil::OPC\_PROP\_ALMLL = 310
- const int ParseUtil::OPC\_PROP\_ALMROC = 311
- const int ParseUtil::OPC\_PROP\_ALMDEV = 312
- const int ParseUtil::OPC\_PROP\_PREC = 8500
- const int ParseUtil::OPC\_PROP\_ZRST = 8510
- const int ParseUtil::OPC\_PROP\_FFST = 8525
- const int ParseUtil::OPC\_PROP\_RECTYPE = 8600
- const int ParseUtil::OPC\_PROP\_INOUT = 8601
- const char \*const ParseUtil::OPC\_PROP\_INPUT = "input"
- const char \*const ParseUtil::OPC\_PROP\_OUTPUT = "output"
- const int ParseUtil::OPC\_PROP\_TSE = 8602
- const int ParseUtil::OPC\_PROP\_PINI = 8603
- const int ParseUtil::OPC\_PROP\_DTYP = 8604
- const int ParseUtil::OPC\_PROP\_SERVER = 8610
- const int ParseUtil::OPC\_PROP\_PLNAME = 8611
- const int ParseUtil::OPC\_PROP\_ALIAS = 8620
- const int ParseUtil::OPC\_PROP\_ALMOSV = 8700
- const int ParseUtil::OPC\_PROP\_ALMZSV = 8701
- const int ParseUtil::OPC\_PROP\_ALMCOSV = 8702
- const int ParseUtil::OPC\_PROP\_ALMUNSV = 8703
- const int ParseUtil::OPC\_PROP\_ALMZRSV = 8710
- const int ParseUtil::OPC\_PROP\_ALMFFSV = 8725
- const int ParseUtil::OPC\_PROP\_ALMHHSV = 8727
- const int ParseUtil::OPC\_PROP\_ALMHHSV = 8728
- const int ParseUtil::OPC\_PROP\_ALMLSV = 8729
- const int ParseUtil::OPC\_PROP\_ALMLLSV = 8730
- const int ParseUtil::OPC\_PROP\_FIELD\_BEG = 8800
- const int ParseUtil::OPC\_PROP\_FIELD\_END = 9000

### 9.18.1 Detailed Description

Header which includes OPC constants.

## 9.19 plcBase.cpp File Reference

```
#include "plcBase.h"
#include <windows.h>
Include dependency graph for plcBase.cpp:
```



## Classes

- struct [plc::scanner\\_thread\\_args](#)

*Scanner thread arguments.*

## Namespaces

- [plc](#)

*Namespace for abstract plc functionality.*

## Functions

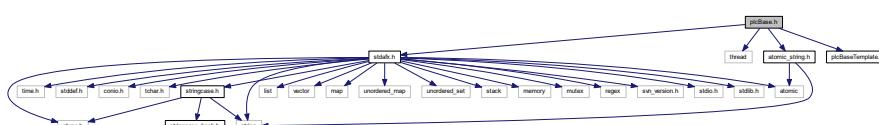
- template<> bool [plc::reset\\_and\\_read](#) (DataValueTypeDef::atomic\_bool &dirty, DataValueTypeDef::type\_<wstring &dest, DataValueTypeDef::atomic\_string \*source)
- Reset and read.*
- template<> bool [plc::write\\_and\\_test](#) (DataValueTypeDef::atomic\_bool &dirty, const DataValueTypeDef::atomic\_bool &read\_pending, DataValueTypeDef::atomic\_bool &valid, DataValueTypeDef::atomic\_wstring \*dest, const DataValueTypeDef::type\_string &source)
- Write and test.*
- VOID CALLBACK [plc::ScannerProc](#) (LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHighValue)
- Scanner thread callback.*
- DWORD WINAPI [plc::scannerThread](#) (scanner\_thread\_args args)
- Scanner thread.*
- [declspec](#) (dllexport) void stopTc(void)
- Stop TwinCAT.*

### 9.19.1 Detailed Description

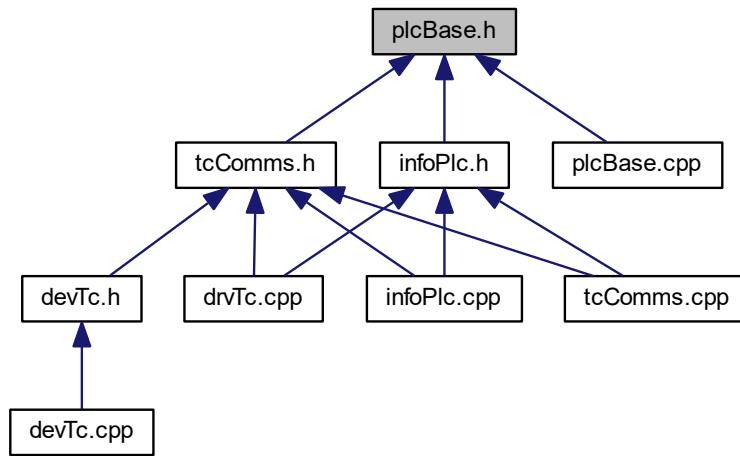
Defines methods for the internal record entry.

## 9.20 plcBase.h File Reference

```
#include "stdafx.h"
#include <thread>
#include "atomic_string.h"
#include "plcBaseTemplate.h"
Include dependency graph for plcBase.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [plc::Interface](#)  
*Abstract interface.*
- struct [plc::DataValueTraits< T >](#)  
*Data value traits.*
- struct [plc::DataValueTypeDef](#)  
*Collection of type definitions.*
- class [plc::DataValue](#)  
*Data value.*
- class [plc::BaseRecord](#)  
*Class for managing a tag/channel.*
- class [plc::BasePLC](#)  
*Base PLC.*
- class [plc::System](#)  
*System to keep track of PLCs.*

## Namespaces

- [plc](#)  
*Namespace for abstract plc functionality.*

## Typedefs

- `typedef std::shared_ptr< BasePLC > plc::BasePLCPtr`  
*Smart pointer to PLC.*
- `typedef std::unique_ptr< Interface > plc::InterfacePtr`  
*Smart pointer to interface.*
- `typedef std::shared_ptr< BaseRecord > plc::BaseRecordPtr`  
*smart pointer to record*
- `typedef std::unordered_map< std::stringcase, BaseRecordPtr > plc::BaseRecordList`  
*list of record*
- `typedef std::map< std::stringcase, BasePLCPtr > plc::BasePLCList`  
*BasePLC map.*

## Enumerations

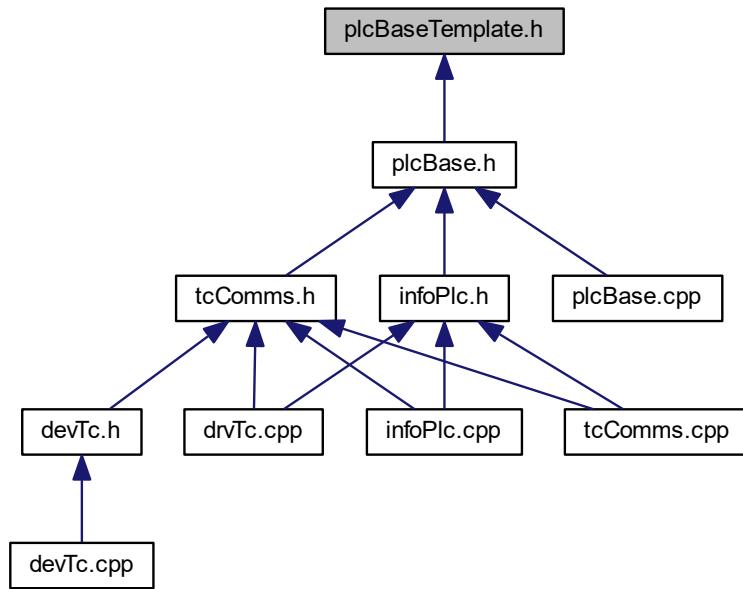
- `enum plc::data_type_enum {`  
`plc::data_type_enum::dtInvalid, plc::data_type_enum::dtBool, plc::data_type_enum::dtInt8, plc::data_type_enum::dtUInt8,`  
`plc::data_type_enum::dtInt16, plc::data_type_enum::dtUInt16, plc::data_type_enum::dtInt32, plc::data_type_enum::dtUInt32,`  
`plc::data_type_enum::dtInt64, plc::data_type_enum::dtUInt64, plc::data_type_enum::dtFloat, plc::data_type_enum::dtDouble,`  
`plc::data_type_enum::dtString, plc::data_type_enum::dtWString, plc::data_type_enum::dtBinary }`  
*Data type enumeration.*
- `enum plc::access_rights_enum {`  
`plc::access_rights_enum::read_only, plc::access_rights_enum::write_only,`  
`plc::access_rights_enum::read_write }`  
*Access rights enum.*

### 9.20.1 Detailed Description

Header which includes abstract base classes for defining an internal record entry for the IOC.

## 9.21 plcBaseTemplate.h File Reference

This graph shows which files directly or indirectly include this file:



## Namespaces

- `plc`

*Namespace for abstract plc functionality.*

## Functions

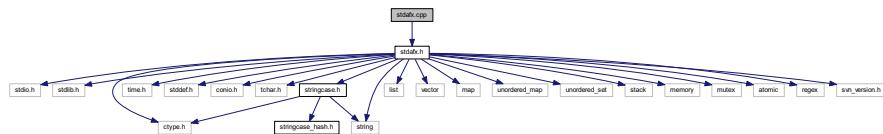
- template<typename T , typename U >  
`bool plc::reset_and_read` (DataValueTypeDef::atomic\_bool &dirty, T &dest, U source)  
*Reset and read.*
- template<typename T >  
`bool plc::reset_and_read` (DataValueTypeDef::atomic\_bool &dirty, T &dest, typename DataValueTraits< T >::traits\_atomic \*source)  
*Reset and read.*
- template<typename T , typename U >  
`bool plc::write_and_test` (DataValueTypeDef::atomic\_bool &dirty, const DataValueTypeDef::atomic\_bool &read\_pending, DataValueTypeDef::atomic\_bool &valid, U dest, const T &source)  
*Write and test.*
- template<typename T >  
`bool plc::write_and_test` (DataValueTypeDef::atomic\_bool &dirty, const DataValueTypeDef::atomic\_bool &read\_pending, DataValueTypeDef::atomic\_bool &valid, typename DataValueTraits< T >::traits\_atomic \*dest, const T &source)  
*Write and test.*

### 9.21.1 Detailed Description

Header which includes templated methods for abstract record class and the DataValue class.

## 9.22 stdafx.cpp File Reference

```
#include "stdafx.h"
Include dependency graph for stdafx.cpp:
```



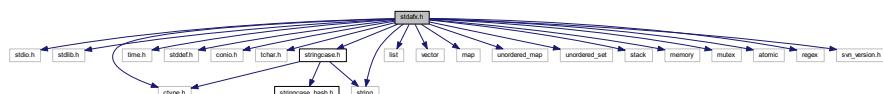
### 9.22.1 Detailed Description

source file that includes just the standard includes EpicsDbGen.pch will be the pre-compiled header stdafx.obj will contain the pre-compiled type information

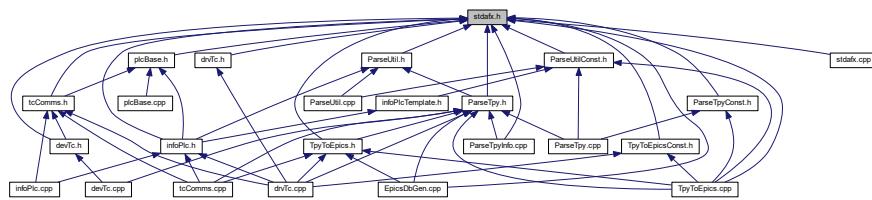
## 9.23 stdafx.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <stddef.h>
#include <conio.h>
#include <tchar.h>
#include <string>
#include "stringcase.h"
#include <list>
#include <vector>
#include <map>
#include <unordered_map>
#include <unordered_set>
#include <stack>
#include <memory>
#include <mutex>
#include <atomic>
#include <regex>
#include "svn_version.h"
```

Include dependency graph for stdafx.h:



This graph shows which files directly or indirectly include this file:

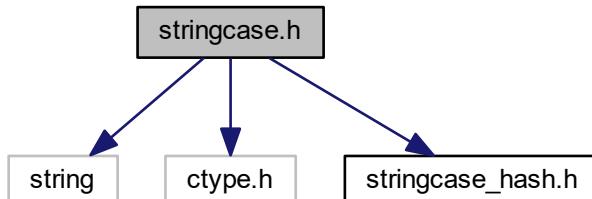


### **9.23.1 Detailed Description**

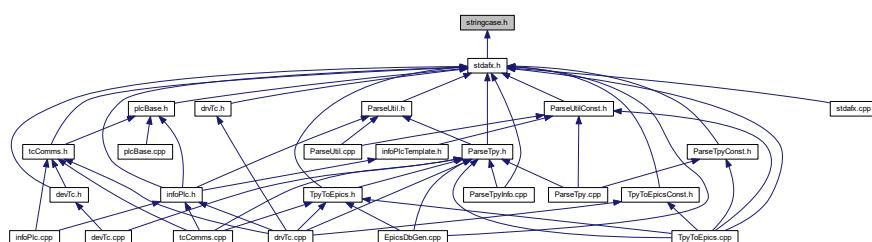
Include file for standard system include files, or project specific include files that are used frequently, but are changed infrequently

## 9.24 stringcase.h File Reference

```
#include <string>
#include <ctype.h>
#include "stringcase_hash.h"
Include dependency graph for stringcase.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- struct `std::case_char_traits`  
*case insensitive traits.*
  - struct `std::case_wchar_traits`  
*case insensitive unicode traits.*

## TypeDefs

- `typedef std::basic_string< char, case_char_traits > std::stringcase`  
*case insensitive string.*
  - `typedef std::basic_string< wchar_t, case_wchar_traits > std::wstringcase`  
*case insensitive string.*

# Functions

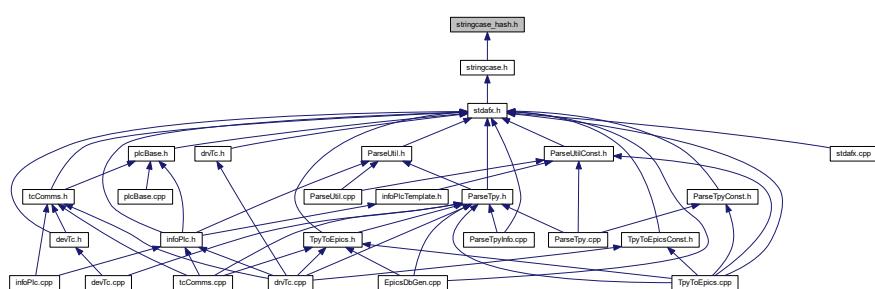
- int `std::strncasecmp` (const char \*s1, const char \*s2, size\_t n)  
*case insensitive compare with maximum length*
  - int `std::wcsncasewcmp` (const wchar\_t \*s1, const wchar\_t \*s2, size\_t n)  
*case insensitive unicode compare with maximum length*
  - void `std::trim_space` (`std::stringcase` &s)
  - void `std::trim_space` (`std::wstringcase` &s)
  - template<class Container , class String , class Predicate >  
void `std::split_string` (Container &output, const String &input, const Predicate &pred, bool trimEmpty=true)  
*Splits a strings.*

### **9.24.1 Detailed Description**

Functions and classes for a case-insensitive string.

## 9.25 stringcase\_hash.h File Reference

This graph shows which files directly or indirectly include this file:



## Classes

- struct `std::std::hash< std::stringcase >`  
`hash for case insensitive string.`
- struct `std::std::hash< std::wstringcase >`  
`hash for case insensitive unicode string.`

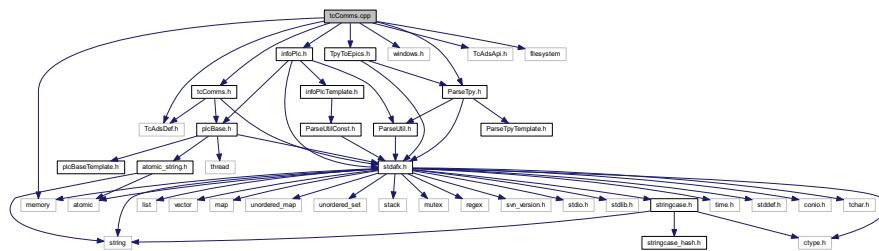
### 9.25.1 Detailed Description

Specialization for case sensitive strings.

## 9.26 tcComms.cpp File Reference

```
#include "tcComms.h"
#include "infoPlc.h"
#include "ParseTpy.h"
#include "windows.h"
#include "TpyToEpics.h"
#include "TcAdsDef.h"
#include "TcAdsApi.h"
#include <memory>
#include <filesystem>
```

Include dependency graph for tcComms.cpp:



## Namespaces

- `TcComms`  
`Namespace for TCat communication.`

## Functions

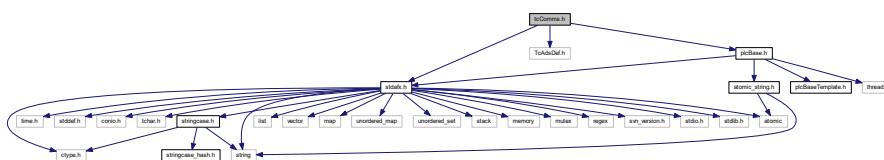
- void `TcComms::errorPrintf (int nErr)`  
`errorPrintf`
- void `__stdcall TcComms::ADSCallback (AmsAddr *pAddr, AdsNotificationHeader *pNotification, unsigned long plcid)`
- void `__stdcall TcComms::RouterCall (long nReason)`

### **9.26.1 Detailed Description**

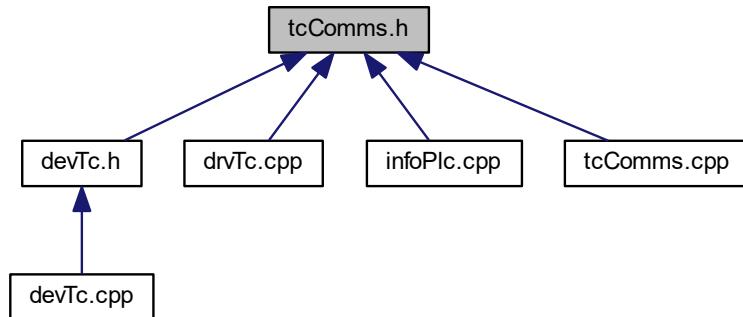
Defines methods for TwinCAT communication.

## 9.27 tcComms.h File Reference

```
#include "stdafx.h"
#include <TcAdsDef.h>
#include "plcBase.h"
Include dependency graph for tcComms.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- struct **TcComms::DataPar**  
*Memory location struct.*
  - class **TcComms::TCatInterface**  
*TCat interface class.*
  - class **TcComms::tcProcWrite**  
*TwinCAT process write requests.*
  - class **TcComms::TcPLC**  
*TwinCAT PLC.*
  - class **TcComms::AmsRouterNotification**  
*AMS Router Notification.*

## Namespaces

- **TcComms**

*Namespace for TCat communication.*

## Variables

- const int **TcComms::MAX\_REQ\_SIZE** = 250000  
*maximum allowed request size (bytes)*
- const int **TcComms::MAX\_SINGLE\_GAP\_SIZE** = 50  
*maximum allowed size (bytes) of a memory gap within continuous request*
- const double **TcComms::MAX\_REL\_GAP** = 0.25  
*(maximum allowed total gap size) / (current request size)*
- const int **TcComms::MIN\_REL\_GAP\_SIZE** = 100  
*minimum allowed relative gap size (bytes)*
- const int **TcComms::default\_scanrate** = 100  
*default PLC TwinCAT scan rate (100ms)*
- const int **TcComms::minimum\_scanrate** = 5  
*minimum PLC TwinCAT scan rate (5ms)*
- const int **TcComms::maximum\_scanrate** = 10000  
*maximum PLC TwinCAT scan rate (10s)*
- const int **TcComms::default\_multiple** = 10  
*default multiple for PLC EPICS scan rate (10)*
- const int **TcComms::minimum\_multiple** = 1  
*minimum multiple for PLC EPICS scan rate (1)*
- const int **TcComms::maximum\_multiple** = 200  
*maximum multiple for PLC EPICS scan rate (200)*

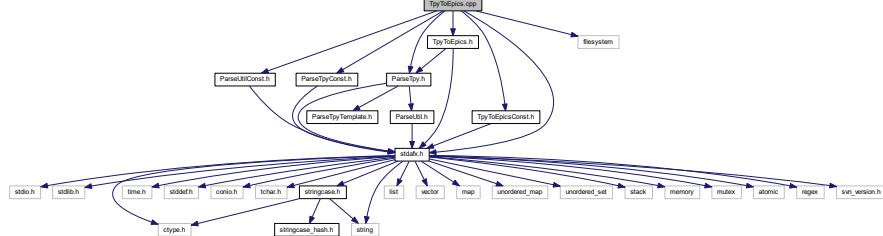
### 9.27.1 Detailed Description

Header which includes classes to interface with the TCat system and manage TCat symbols.

## 9.28 TpyToEpics.cpp File Reference

```
#include "stdafx.h"
#include "ParseUtilConst.h"
#include "ParseTpyConst.h"
#include "ParseTpy.h"
#include "TpyToEpicsConst.h"
#include "TpyToEpics.h"
#include <filesystem>
```

Include dependency graph for TpyToEpics.cpp:



## Namespaces

- [EpicsTpy](#)

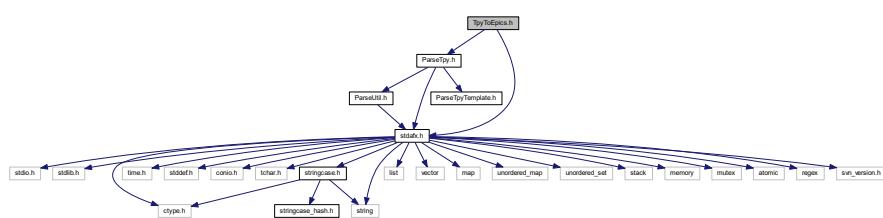
*Namespace for tpy-db conversion.*

### 9.28.1 Detailed Description

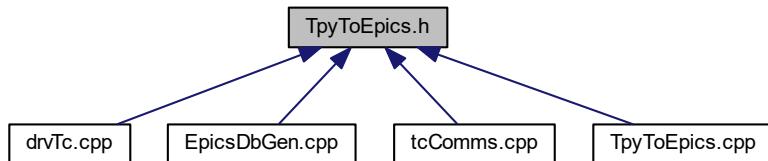
Source for methods that generate EPICs .db file from a .tpy file

## 9.29 TpyToEpics.h File Reference

```
#include "stdaafx.h"
#include "ParseTpy.h"
Include dependency graph for TpyToEpics.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [EpicsTpy::epics\\_conversion](#)  
*Epics conversion.*
- class [EpicsTpy::split\\_io\\_support](#)  
*Split IO support.*
- class [EpicsTpy::multi\\_io\\_support](#)  
*Multiple IO support.*
- class [EpicsTpy::epics\\_list\\_processing](#)  
*List processing.*
- struct [EpicsTpy::macro\\_info](#)

*Macro information.*

- struct [EpicsTpy::macro\\_record](#)

*Macro record.*
- class [EpicsTpy::epics\\_macrofiles\\_processing](#)

*Macro file processing.*
- class [EpicsTpy::epics\\_db\\_processing](#)

*Epics database record processing*

## Namespaces

- [EpicsTpy](#)

*Namespace for tpy-db conversion.*

## Typedefs

- typedef std::vector< macro\_info > [EpicsTpy::macro\\_list](#)
- typedef std::stack< macro\_record > [EpicsTpy::macro\\_stack](#)
- typedef std::unordered\_set< std::stringcase > [EpicsTpy::filename\\_set](#)

## Enumerations

- enum [EpicsTpy::tc\\_epics\\_conv](#) { [EpicsTpy::tc\\_epics\\_conv::no\\_conversion](#), [EpicsTpy::tc\\_epics\\_conv::no\\_dot](#), [EpicsTpy::tc\\_epics\\_conv::ligo\\_std](#), [EpicsTpy::tc\\_epics\\_conv::ligo\\_vac](#) }

*Conversion rules for TC/EPICS.*
- enum [EpicsTpy::case\\_type](#) { [EpicsTpy::case\\_type::preserve](#), [EpicsTpy::case\\_type::upper](#), [EpicsTpy::case\\_type::lower](#) }

*Case conversion rule enum.*
- enum [EpicsTpy::io\\_filestat](#) { [EpicsTpy::io\\_filestat::closed](#), **read**, **write** }

*enum for file io*
- enum [EpicsTpy::listing\\_type](#) { [EpicsTpy::listing\\_type::standard](#), [EpicsTpy::listing\\_type::autoburt](#), [EpicsTpy::listing\\_type::daqini](#) }

*Listing type enum.*
- enum [EpicsTpy::macrofile\\_type](#) { [EpicsTpy::macrofile\\_type::all](#), [EpicsTpy::macrofile\\_type::fields](#), [EpicsTpy::macrofile\\_type::error](#) }
- enum [EpicsTpy::device\\_support\\_type](#) { [EpicsTpy::device\\_support\\_type::opc\\_name](#), [EpicsTpy::device\\_support\\_type::tc\\_name](#) }

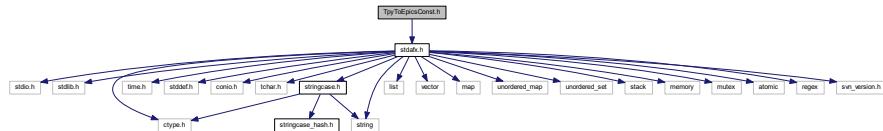
*Device support enum.*

### 9.29.1 Detailed Description

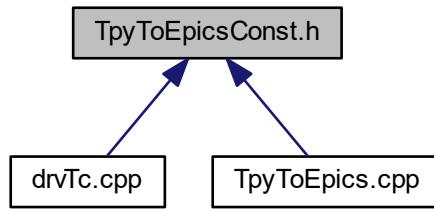
Header which includes classes to convert a parsed TwinCAT tpy into an EPICS database. It includes "ParseTpy.h"

## 9.30 TpyToEpicsConst.h File Reference

```
#include "stdafx.h"
Include dependency graph for TpyToEpicsConst.h:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- [ParseTpy](#)  
*Namespace for parsing.*
- [EpicsTpy](#)  
*Namespace for tpy-db conversion.*

## Variables

- const int `EpicsTpy::MAX_EPICS_CHANNEL` = 54
- const int `EpicsTpy::MAX_EPICS_DESC` = 40
- const int `EpicsTpy::MAX_EPICS_STRING` = 40
- const int `EpicsTpy::MAX_EPICS_UNIT` = 15
- const char \*const `EpicsTpy::EPICS_DB_EGU` = "EGU"
- const char \*const `EpicsTpy::EPICS_DB_DESC` = "DESC"
- const char \*const `EpicsTpy::EPICS_DB_HOPR` = "HOPR"
- const char \*const `EpicsTpy::EPICS_DB_LOPR` = "LOPR"
- const char \*const `EpicsTpy::EPICS_DB_DRVH` = "DRVH"
- const char \*const `EpicsTpy::EPICS_DB_DRVL` = "DRVL"
- const char \*const `EpicsTpy::EPICS_DB_ONAM` = "ONAM"
- const char \*const `EpicsTpy::EPICS_DB_ZNAM` = "ZNAM"
- const char \*const `EpicsTpy::EPICS_DB_PREC` = "PREC"
- const char \*const `EpicsTpy::EPICS_DB_ZRST` [16]

- const char \*const EpicsTpy::EPICS\_DB\_ZRVL [16]
- const char \*const EpicsTpy::EPICS\_DB\_SCAN = "SCAN"
- const char \*const EpicsTpy::EPICS\_DB\_INP = "INP"
- const char \*const EpicsTpy::EPICS\_DB\_OUT = "OUT"
- const char \*const EpicsTpy::EPICS\_DB\_TSE = "TSE"
- const char \*const EpicsTpy::EPICS\_DB\_PINI = "PINI"
- const char \*const EpicsTpy::EPICS\_DB\_DTYP = "DTYP"
- const char \*const EpicsTpy::EPICS\_DB\_OSV = "OSV"
- const char \*const EpicsTpy::EPICS\_DB\_ZSV = "ZSV"
- const char \*const EpicsTpy::EPICS\_DB\_COSV = "COSV"
- const char \*const EpicsTpy::EPICS\_DB\_HIHI = "HIHI"
- const char \*const EpicsTpy::EPICS\_DB\_HIGH = "HIGH"
- const char \*const EpicsTpy::EPICS\_DB\_LOW = "LOW"
- const char \*const EpicsTpy::EPICS\_DB\_LOLO = "LOLO"
- const char \*const EpicsTpy::EPICS\_DB\_HYST = "HYST"
- const char \*const EpicsTpy::EPICS\_DB\_HHSV = "HHSV"
- const char \*const EpicsTpy::EPICS\_DB\_HSV = "HSV"
- const char \*const EpicsTpy::EPICS\_DB\_LSV = "LSV"
- const char \*const EpicsTpy::EPICS\_DB\_LLSV = "LLSV"
- const char \*const EpicsTpy::EPICS\_DB\_NOALARM = "NO\_ALARM"
- const char \*const EpicsTpy::EPICS\_DB\_MINOR = "MINOR"
- const char \*const EpicsTpy::EPICS\_DB\_MAJOR = "MAJOR"
- const char \*const EpicsTpy::EPICS\_DB\_UNSV = "UNSV"
- const char \*const EpicsTpy::EPICS\_DB\_ZRSV [16]
- const char \*const EpicsTpy::EPICS\_DB\_FORBIDDEN []
- const char \*const EpicsTpy::EPICS\_DB\_ALLOWED []
- const char \*const EpicsTpy::EPICS\_DB\_NUMVAL []
- const char \*const EpicsTpy::LIGODAQ\_DATATYPE\_NAME = "datatype"
- const int EpicsTpy::LIGODAQ\_DATATYPE\_FLOAT = 4
- const int EpicsTpy::LIGODAQ\_DATATYPE\_INT32 = 2
- const int EpicsTpy::LIGODAQ\_DATATYPE\_DEFAULT = LIGODAQ\_DATATYPE\_FLOAT
- const char \*const EpicsTpy::LIGODAQ\_UNIT\_NAME = "units"
- const char \*const EpicsTpy::LIGODAQ\_UNIT\_NONE = "none"
- const char \*const EpicsTpy::LIGODAQ\_UNIT\_DEFAULT = LIGODAQ\_UNIT\_NONE
- const char \*const EpicsTpy::LIGODAQ\_INI\_HEADER

### 9.30.1 Detailed Description

Header which includes EPICS constants needed to generate an EPICS db file from a parsed TwinCAT tpy file.



# Index

aaival  
    Device support for TwinCAT/ADS, [17](#)

aooval  
    Device support for TwinCAT/ADS, [17](#)

access\_rights\_enum  
    plc, [94](#)

add  
    plc::BasePLC, [117](#)  
    plc::System, [289](#)  
    TcComms::tcProcWrite, [308](#)

ADSCallback  
    TcComms, [101](#)  
    TcComms::TcPLC, [306](#)

airval  
    Device support for TwinCAT/ADS, [17](#)

aival  
    Device support for TwinCAT/ADS, [17](#)

all  
    EPICS utility functions and classes, [60](#)  
    Parser utility functions and classes, [32](#)

aorval  
    Device support for TwinCAT/ADS, [17](#)

aoaval  
    Device support for TwinCAT/ADS, [17](#)

arraytype  
    TwinCAT tpy file parser, [27](#)

atomic  
    Parser utility functions and classes, [32](#)

atomic\_string.h, [327](#)

autoburst  
    EPICS utility functions and classes, [59](#)

base\_record  
    ParseTpy::base\_record, [113](#), [114](#)

BasePLCList  
    plc, [92](#)

BasePLCPtr  
    plc, [93](#)

BaseRecord  
    plc::BaseRecord, [128](#)

BaseRecordList  
    plc, [93](#)

BaseRecordPtr  
    plc, [93](#)

birval  
    Device support for TwinCAT/ADS, [17](#)

bival  
    Device support for TwinCAT/ADS, [17](#)

borval  
    Device support for TwinCAT/ADS, [17](#)

boval  
    Device support for TwinCAT/ADS, [17](#)

case\_type  
    EPICS utility functions and classes, [58](#)

closed  
    EPICS utility functions and classes, [59](#)

closePort  
    TcComms::TcPLC, [302](#)

compare  
    String functions and classes, [49](#)

compareNamesWoNamespace  
    ParseTpy, [87](#)

cyclesLeft  
    TcComms::TcPLC, [306](#)

daqini  
    EPICS utility functions and classes, [59](#)

data\_enum  
    plc::DataValueTraits< T >, [187](#)

data\_type\_enum  
    plc, [95](#)

WithValue  
    plc::WithValue, [155](#)

dbinfo\_list  
    InfoPlc::Infolnterface, [245](#)

Device driver functions, [22](#)

Device support for TwinCAT/ADS, [13](#)  
    aaival, [17](#)  
    aooval, [17](#)  
    airval, [17](#)  
    aival, [17](#)  
    aorval, [17](#)  
    aoaval, [17](#)  
    birval, [17](#)  
    bival, [17](#)  
    borval, [17](#)  
    boval, [17](#)  
    epics\_record\_enum, [16](#)  
    epics\_record\_enumEnd, [17](#)  
    eventval, [17](#)  
    get\_callback\_queue\_free, [17](#)  
    get\_callback\_queue\_size, [18](#)  
    get\_callback\_queue\_used, [18](#)  
    histogramval, [17](#)  
    invalidval, [17](#)  
    isCallback, [20](#)  
    isPassive, [20](#)  
    linkRecord, [19](#)  
    longinval, [17](#)

longoutval, 17  
 mbbiDirectrval, 17  
 mbbiDirectval, 17  
 mbbirval, 17  
 mbbival, 17  
 mbboDirectrval, 17  
 mbboDirectval, 17  
 mbborval, 17  
 mbboval, 17  
 register\_devsup, 20  
 stringintval, 17  
 stringoutval, 17  
 the\_register\_devsup, 21  
 value\_ait\_type, 21  
 value\_count, 21  
 waveformval, 17  
 device\_support\_type  
     EPICS utility functions and classes, 58  
**DevTc**, 75  
 tc\_regex, 76  
 tcAlias, 77  
 tcInfoPrefix, 78  
 tcList, 78  
 tcLoadRecords, 79  
 tcMacro, 79  
 tcPrintVal, 80  
 tcPrintVals, 81  
 tcSetScanRate, 81  
**devTc.cpp**, 328  
**devTc.h**, 330  
**DevTc::devTcDefIn< RecType >**, 189  
**DevTc::devTcDeflo< RecType >**, 191  
**DevTc::devTcDefOut< RecType >**, 193  
**DevTc::devTcDefWaveformIn< RecType >**, 194  
**DevTc::epics\_record\_traits< RecType >**, 227  
**DevTc::epics\_record\_traits< RecType >::traits\_type**, 320  
**DevTc::epics\_tc\_db\_processing**, 228  
 epics\_tc\_db\_processing, 230  
 operator(), 230  
 patch\_db\_recordnames, 231  
 process\_list, 231  
 process\_lists, 232  
 process\_macro, 232  
 process\_macros, 232  
**DevTc::EpicsInterface**, 233  
**DevTc::register\_devsup**, 276  
**DevTc::tcRegisterToLocShell**, 310  
 dimension  
     TwinCAT tpy file parser, 25  
 dimensions  
     TwinCAT tpy file parser, 26  
 done  
     Info interface classes and functions, 24  
**drvTc.cpp**, 332  
**drvTc.h**, 334  
**dtBinary**  
 plc, 95  
 dtBool  
 plc, 95  
**dtDouble**  
 plc, 95  
**dtFloat**  
 plc, 95  
**dtInt16**  
 plc, 95  
**dtInt32**  
 plc, 95  
**dtInt64**  
 plc, 95  
**dtInt8**  
 plc, 95  
**dtInvalid**  
 plc, 95  
**dtString**  
 plc, 95  
**dtUInt16**  
 plc, 95  
**dtUInt32**  
 plc, 95  
**dtUInt64**  
 plc, 95  
**dtUInt8**  
 plc, 95  
**dtWString**  
 plc, 95  
**enum\_map**  
     TwinCAT tpy file parser, 26  
**enum\_pair**  
     TwinCAT tpy file parser, 26  
**enumtype**  
     TwinCAT tpy file parser, 27  
**EPICS constants**, 61  
 EPICS\_DB\_ALLOWED, 62  
 EPICS\_DB\_COSV, 62  
 EPICS\_DB\_DESC, 62  
 EPICS\_DB\_DRVH, 62  
 EPICS\_DB\_DRVL, 63  
 EPICS\_DB\_DTYP, 63  
 EPICS\_DB\_EGU, 63  
 EPICS\_DB\_FORBIDDEN, 63  
 EPICS\_DB\_HHSV, 64  
 EPICS\_DB\_HIGH, 64  
 EPICS\_DB\_HIHI, 64  
 EPICS\_DB\_HOPR, 64  
 EPICS\_DB\_HSV, 65  
 EPICS\_DB\_HYST, 65  
 EPICS\_DB\_INP, 65  
 EPICS\_DB\_LLSV, 65  
 EPICS\_DB\_LOLO, 66  
 EPICS\_DB\_LOPR, 66  
 EPICS\_DB\_LOW, 66  
 EPICS\_DB\_LSV, 66  
 EPICS\_DB\_MAJOR, 67  
 EPICS\_DB\_MINOR, 67  
 EPICS\_DB\_NOALARM, 67

EPICS\_DB\_NUMVAL, 67  
EPICS\_DB\_ONAM, 68  
EPICS\_DB\_OSV, 68  
EPICS\_DB\_OUT, 68  
EPICS\_DB\_PINI, 68  
EPICS\_DB\_PREC, 69  
EPICS\_DB\_SCAN, 69  
EPICS\_DB\_TSE, 69  
EPICS\_DB\_UNSV, 69  
EPICS\_DB\_ZNAM, 70  
EPICS\_DB\_ZRST, 70  
EPICS\_DB\_ZRSV, 70  
EPICS\_DB\_ZRVL, 70  
EPICS\_DB\_ZSV, 71  
LIGODAQ\_DATATYPE\_DEFAULT, 71  
LIGODAQ\_DATATYPE\_FLOAT, 71  
LIGODAQ\_DATATYPE\_INT32, 71  
LIGODAQ\_DATATYPE\_NAME, 72  
LIGODAQ\_INI\_HEADER, 72  
LIGODAQ\_UNIT\_DEFAULT, 72  
LIGODAQ\_UNIT\_NAME, 72  
LIGODAQ\_UNIT\_NONE, 73  
MAX\_EPICS\_CHANNEL, 73  
MAX\_EPICS\_DESC, 73  
MAX\_EPICS\_STRING, 73  
MAX\_EPICS\_UNIT, 73  
EPICS utility functions and classes, 57  
all, 60  
autoburt, 59  
case\_type, 58  
closed, 59  
daqini, 59  
device\_support\_type, 58  
errors, 60  
fields, 60  
filename\_set, 57  
io\_filestat, 59  
ligo\_std, 60  
ligo\_vac, 60  
listing\_type, 59  
lower, 58  
macro\_list, 58  
macro\_stack, 58  
macrofile\_type, 59  
no\_conversion, 60  
no\_dot, 60  
opc\_name, 59  
preserve, 58  
standard, 59  
tc\_epics\_conv, 60  
tc\_name, 59  
upper, 58  
epics\_conversion  
    EpicsTpy::epics\_conversion, 197, 198  
EPICS\_DB\_ALLOWED  
    EPICS constants, 62  
EPICS\_DB\_COSV  
    EPICS constants, 62  
EPICS\_DB\_DESC  
    EPICS constants, 62  
EPICS\_DB\_DRVH  
    EPICS constants, 62  
EPICS\_DB\_DRVL  
    EPICS constants, 63  
EPICS\_DB\_DTYP  
    EPICS constants, 63  
EPICS\_DB\_EGU  
    EPICS constants, 63  
EPICS\_DB\_FORBIDDEN  
    EPICS constants, 63  
EPICS\_DB\_HHSV  
    EPICS constants, 64  
EPICS\_DB\_HIGH  
    EPICS constants, 64  
EPICS\_DB\_HIHI  
    EPICS constants, 64  
EPICS\_DB\_HOPR  
    EPICS constants, 64  
EPICS\_DB\_HSV  
    EPICS constants, 65  
EPICS\_DB\_HYST  
    EPICS constants, 65  
EPICS\_DB\_INP  
    EPICS constants, 65  
EPICS\_DB\_LLSV  
    EPICS constants, 65  
EPICS\_DB\_LOLO  
    EPICS constants, 66  
EPICS\_DB\_LOPR  
    EPICS constants, 66  
EPICS\_DB\_LOW  
    EPICS constants, 66  
EPICS\_DB\_LSV  
    EPICS constants, 66  
EPICS\_DB\_MAJOR  
    EPICS constants, 67  
EPICS\_DB\_MINOR  
    EPICS constants, 67  
EPICS\_DB\_NOALARM  
    EPICS constants, 67  
EPICS\_DB\_NUMVAL  
    EPICS constants, 67  
EPICS\_DB\_ONAM  
    EPICS constants, 68  
EPICS\_DB\_OSV  
    EPICS constants, 68  
EPICS\_DB\_OUT  
    EPICS constants, 68  
EPICS\_DB\_PINI  
    EPICS constants, 68  
EPICS\_DB\_PREC  
    EPICS constants, 69  
epics\_db\_processing  
    EpicsTpy::epics\_db\_processing, 203  
EPICS\_DB\_SCAN  
    EPICS constants, 69

EPICS\_DB\_TSE  
   EPICS constants, 69

EPICS\_DB\_UNSV  
   EPICS constants, 69

EPICS\_DB\_ZNAM  
   EPICS constants, 70

EPICS\_DB\_ZRST  
   EPICS constants, 70

EPICS\_DB\_ZRSV  
   EPICS constants, 70

EPICS\_DB\_ZRVL  
   EPICS constants, 70

EPICS\_DB\_ZSV  
   EPICS constants, 71

epics\_list\_processing  
   EpicsTpy::epics\_list\_processing, 215

epics\_macrofiles\_processing  
   EpicsTpy::epics\_macrofiles\_processing, 222

epics\_record\_enum  
   Device support for TwinCAT/ADS, 16

epics\_record\_enumEnd  
   Device support for TwinCAT/ADS, 17

epics\_tc\_db\_processing  
   DevTc::epics\_tc\_db\_processing, 230

EpicsDbGen.cpp, 335  
   main, 335

EpicsTpy, 82

EpicsTpy::epics\_conversion, 196  
   epics\_conversion, 197, 198  
    getopt, 199  
    to\_epics, 200

EpicsTpy::epics\_db\_processing, 201  
   epics\_db\_processing, 203  
    getopt, 204  
    my getopt, 205  
    operator(), 206  
    process\_field\_alarm, 208  
    process\_field\_numeric, 209–211  
    process\_field\_string, 212

EpicsTpy::epics\_list\_processing, 213  
   epics\_list\_processing, 215  
    getopt, 216  
    my getopt, 217  
    operator(), 218

EpicsTpy::epics\_macrofiles\_processing, 220  
   epics\_macrofiles\_processing, 222  
    getopt, 223  
    my getopt, 224  
    operator(), 225  
    to\_filename, 226

EpicsTpy::macro\_info, 249

EpicsTpy::macro\_record, 250

EpicsTpy::multi\_io\_support, 255  
   getopt, 258  
   multi\_io\_support, 257

EpicsTpy::split\_io\_support, 280  
   getopt, 283  
   increment, 284

operator=, 285

split\_io\_support, 282, 283

eq  
   String functions and classes, 50

erase  
   plc::BasePLC, 119

errorPrintf  
   TcComms, 102

errors  
   EPICS utility functions and classes, 60

eventval  
   Device support for TwinCAT/ADS, 17

fields  
   EPICS utility functions and classes, 60

filename\_set  
   EPICS utility functions and classes, 57

find  
   plc::BasePLC, 119

for\_each  
   plc::BasePLC, 120  
   plc::System, 290

forever  
   Info interface classes and functions, 24

functionblock  
   TwinCAT tpy file parser, 27

get  
   ParseTpy::ads\_routing\_info, 104, 105  
   ParseUtil::memory\_location, 253  
   ParseUtil::process\_arg\_tc, 274

get\_callback\_queue\_free  
   Device support for TwinCAT/ADS, 17

get\_callback\_queue\_size  
   Device support for TwinCAT/ADS, 18

get\_callback\_queue\_used  
   Device support for TwinCAT/ADS, 18

get\_decoration  
   ParseTpy, 88

get\_full  
   InfoPlc::process\_arg\_info, 271  
   ParseUtil::process\_arg, 269  
   ParseUtil::process\_arg\_tc, 274

get\_infodb  
   InfoPlc::Infolnterface, 243

get\_next  
   plc::BasePLC, 120

get\_pointer  
   ParseTpy, 88

get\_responseBuffer  
   TcComms::TcPLC, 302

get\_symbol\_name  
   InfoPlc::Infolnterface, 244

get\_timestamp\_unix  
   plc::BasePLC, 121

getopt  
   EpicsTpy::epics\_conversion, 199  
   EpicsTpy::epics\_db\_processing, 204  
   EpicsTpy::epics\_list\_processing, 216

EpicsTpy::epics\_macrofiles\_processing, 223  
EpicsTpy::multi\_io\_support, 258  
EpicsTpy::split\_io\_support, 283  
ParseUtil::tag\_processing, 293  
GetValid  
    plc::DataValue, 156

histogramval  
    Device support for TwinCAT/ADS, 17

increment  
    EpicsTpy::split\_io\_support, 284

Info interface classes and functions, 23  
    done, 24  
    forever, 24  
    info\_dbrecord\_type, 23  
    once, 24  
    update\_enum, 23

info\_dbrecord\_type  
    Info interface classes and functions, 23

InfoInterface  
    InfoPlc::InfoInterface, 242

InfoPlc, 84  
infoPlc.cpp, 336  
infoPlc.h, 337  
InfoPlc::InfoInterface, 238  
    dbinfo\_list, 245  
    get\_infodb, 243  
    get\_symbol\_name, 244  
    InfoInterface, 242  
    printVal, 244

InfoPlc::process\_arg\_info, 270  
    get\_full, 271  
    process\_arg\_info, 271

infoPlcTemplate.h, 338

Init  
    plc::DataValue, 156

Interface  
    plc::Interface, 247

InterfacePtr  
    plc, 93

invalidval  
    Device support for TwinCAT/ADS, 17

io\_filestat  
    EPICS utility functions and classes, 59

iocMain.cpp, 339  
    main, 340

isCallback  
    Device support for TwinCAT/ADS, 20

isPassive  
    Device support for TwinCAT/ADS, 20

item\_list  
    TwinCAT tpy file parser, 26

ligo\_std  
    EPICS utility functions and classes, 60

ligo\_vac  
    EPICS utility functions and classes, 60

LIGODAQ\_DATATYPE\_DEFAULT  
    EPICS constants, 71  
LIGODAQ\_DATATYPE\_FLOAT  
    EPICS constants, 71  
LIGODAQ\_DATATYPE\_INT32  
    EPICS constants, 71  
LIGODAQ\_DATATYPE\_NAME  
    EPICS constants, 72

LIGODAQ\_INI\_HEADER  
    EPICS constants, 72

LIGODAQ\_UNIT\_DEFAULT  
    EPICS constants, 72

LIGODAQ\_UNIT\_NAME  
    EPICS constants, 72

LIGODAQ\_UNIT\_NONE  
    EPICS constants, 73

linkRecord  
    Device support for TwinCAT/ADS, 19

listing\_type  
    EPICS utility functions and classes, 59

longinval  
    Device support for TwinCAT/ADS, 17

longoutval  
    Device support for TwinCAT/ADS, 17

lower  
    EPICS utility functions and classes, 58

lt  
    String functions and classes, 51

macro\_list  
    EPICS utility functions and classes, 58

macro\_stack  
    EPICS utility functions and classes, 58

macrofile\_type  
    EPICS utility functions and classes, 59

main  
    EpicsDbGen.cpp, 335  
    iocMain.cpp, 340  
    ParseTpyInfo.cpp, 346

MAX\_EPICS\_CHANNEL  
    EPICS constants, 73

MAX\_EPICS\_DESC  
    EPICS constants, 73

MAX\_EPICS\_STRING  
    EPICS constants, 73

MAX\_EPICS\_UNIT  
    EPICS constants, 73

mbbiDirectrvl  
    Device support for TwinCAT/ADS, 17

mbbiDirectvl  
    Device support for TwinCAT/ADS, 17

mbbirval  
    Device support for TwinCAT/ADS, 17

mbbival  
    Device support for TwinCAT/ADS, 17

mbboDirectrvl  
    Device support for TwinCAT/ADS, 17

mbboDirectvl  
    Device support for TwinCAT/ADS, 17

mbborval

Device support for TwinCAT/ADS, 17  
**mbboval**  
 Device support for TwinCAT/ADS, 17  
**memory\_location**  
 ParseUtil::memory\_location, 252, 253  
**multi\_io\_support**  
 EpicsTpy::multi\_io\_support, 257  
**mygetopt**  
 EpicsTpy::epics\_db\_processing, 205  
 EpicsTpy::epics\_list\_processing, 217  
 EpicsTpy::epics\_macrofiles\_processing, 224  
**mysize**  
 plc::DataValue, 185  
  
**name\_parse**  
 ParseTpy::parserinfo\_type, 266  
**ne**  
 String functions and classes, 51, 52  
**no\_change**  
 Parser utility functions and classes, 31  
**no\_conversion**  
 EPICS utility functions and classes, 60  
**no\_dot**  
 EPICS utility functions and classes, 60  
  
**once**  
 Info interface classes and functions, 24  
**OPC** property constants, 33  
 OPC\_PROP\_ALIAS, 34  
 OPC\_PROP\_ALMAREAS, 34  
 OPC\_PROP\_ALMCONDITION, 34  
 OPC\_PROP\_ALMCOSV, 34  
 OPC\_PROP\_ALMDB, 35  
 OPC\_PROP\_ALMDEV, 35  
 OPC\_PROP\_ALMFFSV, 35  
 OPC\_PROP\_ALMH, 35  
 OPC\_PROP\_ALMHELP, 36  
 OPC\_PROP\_ALMH, 36  
 OPC\_PROP\_ALMHHSV, 36  
 OPC\_PROP\_ALMHSV, 36  
 OPC\_PROP\_ALML, 37  
 OPC\_PROP\_ALMLIMIT, 37  
 OPC\_PROP\_ALMLL, 37  
 OPC\_PROP\_ALMLLSV, 37  
 OPC\_PROP\_ALMLSV, 38  
 OPC\_PROP\_ALMOSV, 38  
 OPC\_PROP\_ALMPRIMARYAREA, 38  
 OPC\_PROP\_ALMROC, 38  
 OPC\_PROP\_ALMSTAT, 38  
 OPC\_PROP\_ALMUNSV, 39  
 OPC\_PROP\_ALMZRSV, 39  
 OPC\_PROP\_ALMZSV, 39  
 OPC\_PROP\_AVI, 39  
 OPC\_PROP\_BGC, 39  
 OPC\_PROP\_BLINK, 40  
 OPC\_PROP\_BMP, 40  
 OPC\_PROP\_CDT, 40  
 OPC\_PROP\_CLOSE, 40  
 OPC\_PROP\_DESC, 40  
  
 OPC\_PROP\_DTYP, 41  
 OPC\_PROP\_FFST, 41  
 OPC\_PROP\_FGC, 41  
 OPC\_PROP\_FIELD\_BEG, 41  
 OPC\_PROP\_FIELD\_END, 42  
 OPC\_PROP\_HIEU, 42  
 OPC\_PROP\_HIRANGE, 42  
 OPC\_PROP\_HTML, 42  
 OPC\_PROP\_INOUT, 43  
 OPC\_PROP\_INPUT, 43  
 OPC\_PROP\_LOEU, 43  
 OPC\_PROP\_LORANGE, 43  
 OPC\_PROP\_OPEN, 44  
 OPC\_PROP\_OUTPUT, 44  
 OPC\_PROP\_PINI, 44  
 OPC\_PROP\_PLNAME, 44  
 OPC\_PROP\_PREC, 45  
 OPC\_PROP\_QUALITY, 45  
 OPC\_PROP\_RECTYPE, 45  
 OPC\_PROP\_RIGHTS, 45  
 OPC\_PROP\_SCANRATE, 46  
 OPC\_PROP\_SERVER, 46  
 OPC\_PROP SND, 46  
 OPC\_PROP\_TIME, 46  
 OPC\_PROP\_TIMEZONE, 46  
 OPC\_PROP\_TSE, 47  
 OPC\_PROP\_UNIT, 47  
 OPC\_PROP\_VALUE, 47  
 OPC\_PROP\_ZRST, 47  
**opc\_enum**  
 Parser utility functions and classes, 31  
**opc\_name**  
 EPICS utility functions and classes, 59  
**OPC\_PROP\_ALIAS**  
 OPC property constants, 34  
**OPC\_PROP\_ALMAREAS**  
 OPC property constants, 34  
**OPC\_PROP\_ALMCONDITION**  
 OPC property constants, 34  
**OPC\_PROP\_ALMCOSV**  
 OPC property constants, 34  
**OPC\_PROP\_ALMDB**  
 OPC property constants, 35  
**OPC\_PROP\_ALMDEV**  
 OPC property constants, 35  
**OPC\_PROP\_ALMFFSV**  
 OPC property constants, 35  
**OPC\_PROP\_ALMH**  
 OPC property constants, 35  
**OPC\_PROP\_ALMHELP**  
 OPC property constants, 36  
**OPC\_PROP\_ALMH**  
 OPC property constants, 36  
**OPC\_PROP\_ALMHHSV**  
 OPC property constants, 36  
**OPC\_PROP\_ALMHSV**  
 OPC property constants, 36  
**OPC\_PROP\_ALML**

OPC property constants, 37  
OPC\_PROP\_ALMLIMIT  
    OPC property constants, 37  
OPC\_PROP\_ALMLL  
    OPC property constants, 37  
OPC\_PROP\_ALMLLSV  
    OPC property constants, 37  
OPC\_PROP\_ALMLSV  
    OPC property constants, 38  
OPC\_PROP\_ALMOSV  
    OPC property constants, 38  
OPC\_PROP\_ALMPRIMARYAREA  
    OPC property constants, 38  
OPC\_PROP\_ALMROC  
    OPC property constants, 38  
OPC\_PROP\_ALMSTAT  
    OPC property constants, 38  
OPC\_PROP\_ALMUNSV  
    OPC property constants, 39  
OPC\_PROP\_ALMZRSV  
    OPC property constants, 39  
OPC\_PROP\_ALMZSV  
    OPC property constants, 39  
OPC\_PROP\_AVI  
    OPC property constants, 39  
OPC\_PROP\_BGC  
    OPC property constants, 39  
OPC\_PROP\_BLINK  
    OPC property constants, 40  
OPC\_PROP\_BMP  
    OPC property constants, 40  
OPC\_PROP\_CDT  
    OPC property constants, 40  
OPC\_PROP\_CLOSE  
    OPC property constants, 40  
OPC\_PROP\_DESC  
    OPC property constants, 40  
OPC\_PROP\_DTYP  
    OPC property constants, 41  
OPC\_PROP\_FFST  
    OPC property constants, 41  
OPC\_PROP\_FGC  
    OPC property constants, 41  
OPC\_PROP\_FIELD\_BEG  
    OPC property constants, 41  
OPC\_PROP\_FIELD\_END  
    OPC property constants, 42  
OPC\_PROP\_HIEU  
    OPC property constants, 42  
OPC\_PROP\_HIRANGE  
    OPC property constants, 42  
OPC\_PROP\_HTML  
    OPC property constants, 42  
OPC\_PROP\_INOUT  
    OPC property constants, 43  
OPC\_PROP\_INPUT  
    OPC property constants, 43  
OPC\_PROP\_LOEU  
    OPC property constants, 43  
OPC property constants, 43  
OPC\_PROP\_LORANGE  
    OPC property constants, 43  
OPC\_PROP\_OPEN  
    OPC property constants, 44  
OPC\_PROP\_OUTPUT  
    OPC property constants, 44  
OPC\_PROP\_PINI  
    OPC property constants, 44  
OPC\_PROP\_PLNAME  
    OPC property constants, 44  
OPC\_PROP\_PREC  
    OPC property constants, 45  
OPC\_PROP\_QUALITY  
    OPC property constants, 45  
OPC\_PROP\_RECTYPE  
    OPC property constants, 45  
OPC\_PROP\_RIGHTS  
    OPC property constants, 45  
OPC\_PROP\_SCANRATE  
    OPC property constants, 46  
OPC\_PROP\_SERVER  
    OPC property constants, 46  
OPC\_PROP SND  
    OPC property constants, 46  
OPC\_PROP\_TIME  
    OPC property constants, 46  
OPC\_PROP\_TIMEZONE  
    OPC property constants, 46  
OPC\_PROP\_TSE  
    OPC property constants, 47  
OPC\_PROP\_UNIT  
    OPC property constants, 47  
OPC\_PROP\_VALUE  
    OPC property constants, 47  
OPC\_PROP\_ZRST  
    OPC property constants, 47  
operator()  
    DevTc::epics\_tc\_db\_processing, 230  
    EpicsTpy::epics\_db\_processing, 206  
    EpicsTpy::epics\_list\_processing, 218  
    EpicsTpy::epics\_macrofiles\_processing, 225  
    std::std::hash< std::stringcase >, 236  
    std::std::hash< std::wstringcase >, 237  
operator=()  
    EpicsTpy::split\_io\_support, 285  
    std::atomic\_string< stringT >, 111  
optarg  
    ParseUtil::optarg, 262  
optimizeRequests  
    TcComms::TcPLC, 303  
parse  
    ParseUtil::optarg, 262  
parse\_finish  
    ParseTpy::tpy\_file, 312  
Parser utility functions and classes, 30  
    all, 32  
    atomic, 32

no\_change, 31  
 opc\_enum, 31  
 process\_tag\_enum, 32  
 process\_type\_enum, 32  
 property\_el, 30  
 property\_map, 31  
 pt\_binary, 32  
 pt\_bool, 32  
 pt\_enum, 32  
 pt\_int, 32  
 pt\_invalid, 32  
 pt\_real, 32  
 pt\_string, 32  
 publish, 31  
 replacement\_table, 31  
 silent, 31  
 structured, 32  
**ParseTpy**, 85  
 compareNamesWoNamespace, 87  
 get\_decoration, 88  
 get\_pointer, 88  
**ParseTpy.cpp**, 340  
**ParseTpy.h**, 341  
**ParseTpy::ads\_routing\_info**, 103  
 get, 104, 105  
 set, 105  
**ParseTpy::base\_record**, 111  
 base\_record, 113, 114  
**ParseTpy::compiler\_info**, 150  
**ParseTpy::item\_record**, 248  
**ParseTpy::parserinfo\_type**, 263  
 name\_parse, 266  
**ParseTpy::project\_record**, 275  
**ParseTpy::symbol\_record**, 286  
**ParseTpy::tpy\_file**, 311  
 parse\_finish, 312  
 process\_array, 313  
 process\_symbols, 314  
 process\_type\_tree, 315, 317, 319  
**ParseTpy::type\_map**, 321  
**ParseTpy::type\_record**, 322  
**ParseTpyConst.h**, 343  
**ParseTpyInfo.cpp**, 345  
 main, 346  
**ParseTpyTemplate.h**, 347  
**ParseUtil**, 88  
**ParseUtil.cpp**, 347  
**ParseUtil.h**, 348  
**ParseUtil::bit\_location**, 146  
**ParseUtil::memory\_location**, 251  
 get, 253  
 memory\_location, 252, 253  
 set, 254  
 set\_section, 254  
**ParseUtil::opc\_list**, 259  
**ParseUtil::optarg**, 260  
 optarg, 262  
 parse, 262  
**ParseUtil::process\_arg**, 266  
 get\_full, 269  
 process\_arg, 267  
**ParseUtil::process\_arg\_tc**, 272  
 get, 274  
 get\_full, 274  
 process\_arg\_tc, 273  
**ParseUtil::replacement\_rules**, 278  
**ParseUtil::tag\_processing**, 291  
 getopt, 293  
 tag\_processing, 292, 293  
**ParseUtil::variable\_name**, 324  
**ParseUtilConst.h**, 349  
**patch\_db\_recordnames**  
 DevTc::epics\_tc\_db\_processing, 231  
**plc**, 91  
 access\_rights\_enum, 94  
 BasePLCList, 92  
 BasePLCPtr, 93  
 BaseRecordList, 93  
 BaseRecordPtr, 93  
 data\_type\_enum, 95  
 dtBinary, 95  
 dtBool, 95  
 dtDouble, 95  
 dtFloat, 95  
 dtInt16, 95  
 dtInt32, 95  
 dtInt64, 95  
 dtInt8, 95  
 dtInvalid, 95  
 dtString, 95  
 dtUInt16, 95  
 dtUInt32, 95  
 dtUInt64, 95  
 dtUInt8, 95  
 dtWString, 95  
 InterfacePtr, 93  
 read\_only, 95  
 read\_write, 95  
 reset\_and\_read, 95, 96  
 ScannerProc, 96  
 scannerThread, 97  
 write\_and\_test, 98, 99  
 write\_only, 95  
**plc::BasePLC**, 114  
 add, 117  
 erase, 119  
 find, 119  
 for\_each, 120  
 get\_next, 120  
 get\_timestamp\_unix, 121  
 plc\_data\_set\_valid, 122  
 printRecord, 123  
 records, 124  
 reserve, 123  
 start, 123  
 user\_data\_set\_valid, 123

plc::BaseRecord, 125  
BaseRecord, 128  
PlcGetValid, 128  
PlcPush, 129  
PlcRead, 130, 131  
PlcReadBinary, 132  
PlcSetValid, 133  
PlcWrite, 134, 135  
PlcWriteBinary, 136  
UserGetValid, 137  
UserPush, 137  
UserRead, 139–141  
UserReadBinary, 142  
UserSetValid, 142  
UserWrite, 143–145  
UserWriteBinary, 145  
plc::DataValue, 152  
    DataValue, 155  
    GetValid, 156  
    Init, 156  
    mysize, 185  
    PlcGetValid, 157  
    PlcRead, 158–160  
    PlcReadBinary, 160  
    PlcSetValid, 161  
    PlcWrite, 162–164  
    PlcWriteBinary, 165  
    Read, 166, 167, 169  
    ReadBinary, 170  
    SetValid, 171  
    UserGetValid, 172  
    UserRead, 172–174  
    UserReadBinary, 175  
    UserSetValid, 176  
    UserWrite, 176–178  
    UserWriteBinary, 179  
    Write, 180–183  
    WriteBinary, 184  
plc::DataValueTraits< T >, 185  
    data\_enum, 187  
plc::DataValueTypeDef, 187  
plc::Interface, 245  
    Interface, 247  
    printVal, 247  
plc::scanner\_thread\_args, 279  
plc::System, 288  
    add, 289  
    for\_each, 290  
    printVal, 290  
plc\_data\_set\_valid  
    plc::BasePLC, 122  
plcBase.cpp, 351  
plcBase.h, 352  
plcBaseTemplate.h, 355  
PlcGetValid  
    plc::BaseRecord, 128  
    plc::DataValue, 157  
PlcPush  
    plc::BaseRecord, 129  
PlcRead  
    plc::BaseRecord, 130, 131  
    plc::DataValue, 158–160  
PlcReadBinary  
    plc::BaseRecord, 132  
    plc::DataValue, 160  
PlcSetValid  
    plc::BaseRecord, 133  
    plc::DataValue, 161  
PlcWrite  
    plc::BaseRecord, 134, 135  
    plc::DataValue, 162–164  
PlcWriteBinary  
    plc::BaseRecord, 136  
    plc::DataValue, 165  
preserve  
    EPICS utility functions and classes, 58  
printRecord  
    plc::BasePLC, 123  
    TcComms::TcPLC, 303  
printVal  
    InfoPlc::InfoInterface, 244  
    plc::Interface, 247  
    plc::System, 290  
    TcComms::TCatInterface, 298  
process\_arg  
    ParseUtil::process\_arg, 267  
process\_arg\_info  
    InfoPlc::process\_arg\_info, 271  
process\_arg\_tc  
    ParseUtil::process\_arg\_tc, 273  
process\_array  
    ParseTpy::tpy\_file, 313  
process\_field\_alarm  
    EpicsTpy::epics\_db\_processing, 208  
process\_field\_numeric  
    EpicsTpy::epics\_db\_processing, 209–211  
process\_field\_string  
    EpicsTpy::epics\_db\_processing, 212  
process\_list  
    DevTc::epics\_tc\_db\_processing, 231  
process\_lists  
    DevTc::epics\_tc\_db\_processing, 232  
process\_macro  
    DevTc::epics\_tc\_db\_processing, 232  
process\_macros  
    DevTc::epics\_tc\_db\_processing, 232  
process\_symbols  
    ParseTpy::tpy\_file, 314  
process\_tag\_enum  
    Parser utility functions and classes, 32  
process\_type\_enum  
    Parser utility functions and classes, 32  
process\_type\_tree  
    ParseTpy::tpy\_file, 315, 317, 319  
property\_el  
    Parser utility functions and classes, 30

**property\_map**  
 Parser utility functions and classes, 31  
**pt\_binary**  
 Parser utility functions and classes, 32  
**pt\_bool**  
 Parser utility functions and classes, 32  
**pt\_enum**  
 Parser utility functions and classes, 32  
**pt\_int**  
 Parser utility functions and classes, 32  
**pt\_invalid**  
 Parser utility functions and classes, 32  
**pt\_real**  
 Parser utility functions and classes, 32  
**pt\_string**  
 Parser utility functions and classes, 32  
**publish**  
 Parser utility functions and classes, 31  
  
**Read**  
 plc::DataValue, 166, 167, 169  
**read\_only**  
 plc, 95  
**read\_ptr**  
 TcComms::tcProcWrite, 309  
**read\_write**  
 plc, 95  
**ReadBinary**  
 plc::DataValue, 170  
**records**  
 plc::BasePLC, 124  
**register\_devsup**  
 Device support for TwinCAT/ADS, 20  
**replacement\_table**  
 Parser utility functions and classes, 31  
**req**  
 TcComms::tcProcWrite, 310  
**reserve**  
 plc::BasePLC, 123  
**reset\_and\_read**  
 plc, 95, 96  
**RouterCall**  
 TcComms, 102  
 TcComms::AmsRouterNotification, 107  
  
**ScannerProc**  
 plc, 96  
**scannerThread**  
 plc, 97  
**set**  
 ParseTpy::ads\_routing\_info, 105  
 ParseUtil::memory\_location, 254  
**set\_addr**  
 TcComms::TcPLC, 304  
**set\_ads\_state**  
 TcComms::TcPLC, 304  
**set\_section**  
 ParseUtil::memory\_location, 254  
**SetValid**

plc::DataValue, 171  
**silent**  
 Parser utility functions and classes, 31  
**simple**  
 TwinCAT tpy file parser, 27  
**split\_io\_support**  
 EpicsTpy::split\_io\_support, 282, 283  
**split\_string**  
 String functions and classes, 52  
**standard**  
 EPICS utility functions and classes, 59  
**start**  
 plc::BasePLC, 123  
**std::atomic< string >**, 107  
**std::atomic< wstring >**, 108  
**std::atomic\_string< stringT >**, 110  
 operator=, 111  
**std::case\_char\_traits**, 147  
**std::case\_wchar\_traits**, 149  
**std::std::hash< std::stringcase >**, 236  
 operator(), 236  
**std::std::hash< std::wstringcase >**, 237  
 operator(), 237  
**stdafx.cpp**, 356  
**stdafx.h**, 356  
**String functions and classes**, 48  
 compare, 49  
 eq, 50  
 lt, 51  
 ne, 51, 52  
 split\_string, 52  
 stringcase, 48  
 strncasecmp, 53  
 trim\_space, 54  
 wcsncasewcmp, 55  
 wstringcase, 48  
**stringcase**  
 String functions and classes, 48  
**stringcase.h**, 357  
**stringcase\_hash.h**, 358  
**stringintval**  
 Device support for TwinCAT/ADS, 17  
**stringoutval**  
 Device support for TwinCAT/ADS, 17  
**strncasecmp**  
 String functions and classes, 53  
**structtype**  
 TwinCAT tpy file parser, 27  
**structured**  
 Parser utility functions and classes, 32  
**symbol\_list**  
 TwinCAT tpy file parser, 26  
**syminfo\_processing**, 287  
  
**tag\_processing**  
 ParseUtil::tag\_processing, 292, 293  
**tc\_epics\_conv**  
 EPICS utility functions and classes, 60  
**tc\_name**

EPICS utility functions and classes, 59  
tc\_regex  
    DevTc, 76  
tcAlias  
    DevTc, 77  
TCatInterface  
    TcComms::TCatInterface, 297  
TcComms, 100  
    ADScallback, 101  
    errorPrintf, 102  
    RouterCall, 102  
tcComms.cpp, 359  
tcComms.h, 360  
TcComms::AmsRouterNotification, 106  
    RouterCall, 107  
TcComms::DataPar, 151  
TcComms::TCatInterface, 295  
    printVal, 298  
    TCatInterface, 297  
TcComms::TcPLC, 299  
    ADScallback, 306  
    closePort, 302  
    cyclesLeft, 306  
    get\_responseBuffer, 302  
    optimizeRequests, 303  
    printRecord, 303  
    set\_addr, 304  
    set\_ads\_state, 304  
    update\_scanner, 305  
TcComms::tcProcWrite, 307  
    add, 308  
    read\_ptr, 309  
    req, 310  
tcInfoPrefix  
    DevTc, 78  
tcList  
    DevTc, 78  
tcLoadRecords  
    DevTc, 79  
tcMacro  
    DevTc, 79  
tcPrintVal  
    DevTc, 80  
tcPrintVals  
    DevTc, 81  
tcSetScanRate  
    DevTc, 81  
the\_register\_devsup  
    Device support for TwinCAT/ADS, 21  
to\_epics  
    EpicsTpy::epics\_conversion, 200  
to\_filename  
    EpicsTpy::epics\_macrofiles\_processing, 226  
TpyToEpics.cpp, 361  
TpyToEpics.h, 362  
TpyToEpicsConst.h, 364  
trim\_space  
    String functions and classes, 54  
TwinCAT read/write scanning, 56  
TwinCAT tpy file parser, 25  
    arraytype, 27  
    dimension, 25  
    dimensions, 26  
    enum\_map, 26  
    enum\_pair, 26  
    enumtype, 27  
    functionblock, 27  
    item\_list, 26  
    simple, 27  
    structtype, 27  
    symbol\_list, 26  
    type\_enum, 27  
    type\_multipmap, 27  
    unknown, 27  
type\_enum  
    TwinCAT tpy file parser, 27  
type\_multipmap  
    TwinCAT tpy file parser, 27  
unknown  
    TwinCAT tpy file parser, 27  
update\_enum  
    Info interface classes and functions, 23  
update\_scanner  
    TcComms::TcPLC, 305  
upper  
    EPICS utility functions and classes, 58  
user\_data\_set\_valid  
    plc::BasePLC, 123  
UserGetValid  
    plc::BaseRecord, 137  
    plc::DataValue, 172  
UserPush  
    plc::BaseRecord, 137  
UserRead  
    plc::BaseRecord, 139–141  
    plc::DataValue, 172–174  
UserReadBinary  
    plc::BaseRecord, 142  
    plc::DataValue, 175  
UserSetValid  
    plc::BaseRecord, 142  
    plc::DataValue, 176  
UserWrite  
    plc::BaseRecord, 143–145  
    plc::DataValue, 176–178  
UserWriteBinary  
    plc::BaseRecord, 145  
    plc::DataValue, 179  
value\_ait\_type  
    Device support for TwinCAT/ADS, 21  
value\_count  
    Device support for TwinCAT/ADS, 21  
waveformval  
    Device support for TwinCAT/ADS, 17

wcsncasewcmp  
String functions and classes, 55

Write  
plc::DataValue, 180–183

write\_and\_test  
plc, 98, 99

write\_only  
plc, 95

WriteBinary  
plc::DataValue, 184

wstringcase  
String functions and classes, 48

XML tpy file constants, 28